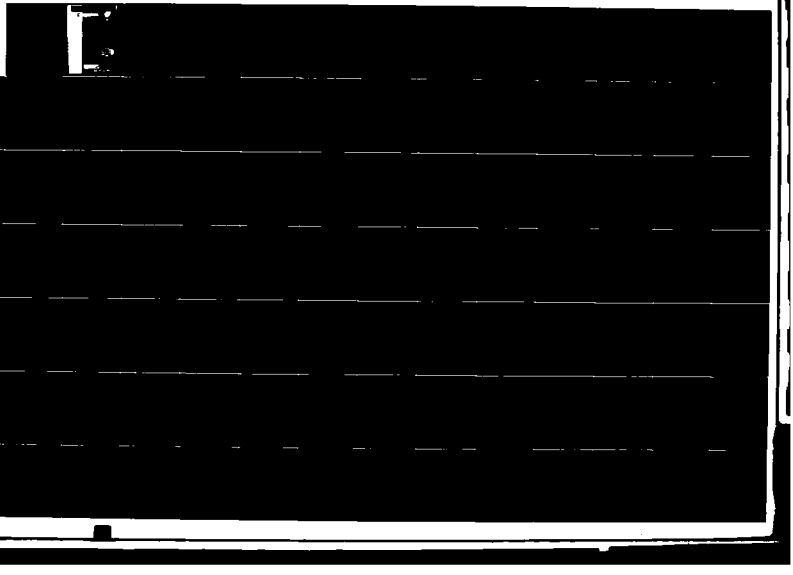
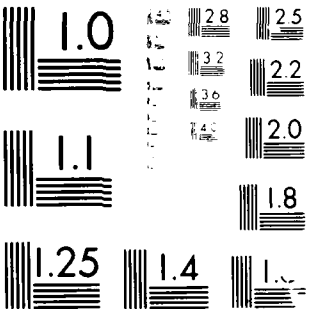
AD-A191 923 THE ADAPTIVE MANEUVERING LOGIC IN TANK WARFARE SIMULATION(U) DECISION SCIENCE INC SAN DIEGO CA A J QUEENS ET AL. 20 MAY 82 DST-82-413-F 1/2

UNCLASSIFIED MDA903-81-C-0589 F/G 12/5 NL




1.0
2.8
2.5
3.2
2.2
3.6
2.0
1.1
1.8
1.25
1.4

[redacted]

THE ADAPTIVE MANEUVERING LOGIC IN TANK WARFARE SIMULATION

FINAL REPORT

[redacted]

DECISION SCIENCE, INC.
4901 MORENA BOULEVARD
SAN DIEGO, CALIFORNIA
92117 (714) 273-2922

DSI-82-413-F

THE ADAPTIVE MANEUVERING LOGIC IN TANK WARFARE SIMULATION

FINAL REPORT

Sponsored by:

Defense Advanced Research Project Agency (DoD)
Under Contract No. MDA903-81-C-0509
Issued by Department of Army,
Defense Supply Service-Washington
Washington, DC 20310

Submitted to:

Major Jack Thorpe, USAF
Defense Advanced Research Project Agency
Cybernetics Technology Division
1400 Wilson Boulevard
Arlington, VA 22209

Prepared by:

A. J. Owens, Ph.D.
R. F. Stalder, Jr.

May 20, 1982

TABLE OF CONTENTS

INTRODUCTION

This Final Report describes the research effort performed by Decision Science, Inc., under Contract No. MDA903-81-C-0509, directed toward the design and development of a computer program for realistic, intelligently interactive tank warfare simulation. Included in the report are the functional description, flow diagrams and preliminary software specifications for such a program. The program offers a unique forward development in the realism of computer generated simulation in that the simulated tank it controls operates in an intelligently interactive manner in opposition to a human controlled tank. That is, the computer program selects moves and countermoves not according to a canned or predetermined routine but according to a rational evaluative criteria in response to action taken by the human trainee opponent.

Drawing on experience gained in the development of the Adaptive Maneuvering Logic (AML) program for air-to-air and naval combat simulation, the first phase of this nine-month contract effort focused on the adaptation of that same underlying concept to the maneuver characteristics and operational options of the modern tank in a ground combat environment. With the professional guidance provided through visits and consultation with tank warfare experts at the U.S. Army School, Fort Knox and at Army Headquarters,

the Pentagon, a Valuated State Space (VSS) was constructed defining the purpose of the tank commander in combat--analogous to the purpose of a fighter pilot in air-to-air combat although complicated by considerations of the terrestrial environment which figures as prominently as the relative position and characteristics of the opposing threat(s). As described in the First Progress Report, included here as Appendix A, the VSS then became the basis for evaluating and selecting optimum positions for the AML controlled tank vis-a-vis the opposing (trainee controlled) tank in light of mission, terrain features, and threat environment.

Once a methodology for evaluating and selecting optimum destination positions had been formulated, the emphasis shifted to the development and testing of an algorithm for determining the "best" path for traversing from the present position to the selected destination. The Second Progress Report, contained herein as Appendix B, describes the evolution of the algorithm and provides an illustrative example of its application. With a view toward moderating computer computational requirements a second algorithm, essentially the same in principle but modified in approach, was developed as described in Appendix C. Both algorithms have been shown equal to the task of ferreting out the "best" path from among all possible paths between a given starting position (i.e., the AML tank's "present position") and an intended destination (i.e., the selected optimum

position vis-a-vis the opposing tank). Preliminary examination indicates that each has advantageous features and selection of which algorithm should be incorporated in the final program can be best determined during the programming phase. Neither appears to impose computational demands beyond the capacity of modern computers of a size and cost to be compatable with relatively small scale, moderately priced training devices.

The concluding phase of the contract period was devoted to defining functional flow sequences and programming requirements as described in the body of this report. It should be noted that although the work to date has necessarily been confined to first solving the one-on-one tank engagement problem, it has progressed with the longer range objective clearly in view of its subsequent expansion to a multiple-tank, multiple-threat combat engagement. Similarly, the Valuated State Space governing the responses of the AML tank was constructed according to the best available tank warfare expertise--U.S. Army armor specialists at Fort Knox and at Army Headquarters. The AML tank is, therefore, at this time modeled in the mirror image of the U.S. Army. However, in the course of developing the present model, the groundwork has been laid with knowledgeable members within the intelligence community with a view toward subsequent modification of the Valuated State Space to reflect the purpose of the Soviet rather than the U.S. Army Tank Commander.

DISCUSSION

SECTION I - OVERVIEW

Decision Science, Inc.'s approach to tank warfare simulation is an outgrowth of work successfully accomplished in the area of air combat and submarine warfare simulation. This approach is based on the Adaptive Maneuvering Logic concept wherein an interactive, intelligent logic was developed to control a tank, or a platoon of tanks, operating under the control of a platoon commander. This as opposed either to control based on heuristics gained from discussions with tank commanders or a canned scenario.

Work was developed guided by the concept "What is the approach taken by an intelligent tank commander?" Fundamental to that question, of course, is the determination of what is a desirable or an undesirable position for the tank. It quickly became evident that the question could only be answered relative to the position of another opponent(s). That is, there is safety behind a hill from an opponent if, and only if, the hill intervenes. If both tanks are on the same side of the hill and are visible one to the other, then the hill is of little import. It was decided that the evaluation of the desirability of a position relative to an opponent at another position could best be accomplished through use of a Valuated State Space (VSS). The First Progress Report, included here for ease of

reference as Appendix A, describes the underlying concept of the hierarchic Valuated State Space (VSS) and contains an illustrative example of a VSS constructed to define in a broad sense the purpose of a tank commander in combat.

A more specific VSS, particularized for the scenario and combat engagement area envisioned as a representative model for developing an Adaptive Maneuvering Logic program for tank warfare simulation, is shown in Section II.

With an evaluation means at hand, the question remained as to how to locate what is a desirable position, called here a candidate position. Two approaches came to mind: (1) an exhaustive evaluation of "all" possible positions and (2) an evaluation of possible positions taken from a previously specified list. Certainly, for any given position that a tank may occupy within a depicted combat engagement area there are certain positions which stand out as having high potential value from the standpoint of an opposing tank. These can be pre-identified by grid square and placed in a prioritized list. Obviously, some loss in optimality is to be expected but not so great as to degrade a worthy opponent into an unworthy one or to so limit the optional moves as to detract significantly from the intelligently interactive response characteristics of the AML opponent.

Assuming the "best" candidate position has been selected from those several pre-identified as having high potential value, the problem remains of determining a path

from the present position to that desired position. Two approaches to the problem were explored resulting in two algorithms.

Appendix B describes an algorithm for determining the "best" paths to a destination point (i.e, the selected candidate position) from points on an incrementally expanding perimeter about that destination point (paths considered are restricted to positions on and within the area considered).

Appendix C describes an algorithm whereby the "best" or maximum utility path emanating from the present position is extended until it is not the path of maximum utility. The current maximum utility path is then extended until it is no longer the maximum utility path. In the first case, the algorithm continues until the present position is encountered and, in the second case, the algorithm continues until the destination is encountered. There may even be a higher utility, more circuitous path to be discovered if the area of concern is increased.

To summarize, discussed so far are procedures for (1) evaluating a position, (2) locating candidate (desirable) positions, and (3) determining a path from a present position to a candidate position.

Since the desired position and its associated path are determined relative to a present position of the opponent tank, they represent a "snap-shot" solution valid only for that instant in time or for as long as the opponent tank

remains in the present position. As the opponent tank moves to a new "present position," the process of selecting a new desired position for the AML tank and determining a best path to that position must be repeated. During this process the AML tank must continue on its previously determined path until the new path is determined and the AML tank redirected along it (at most, a matter of a few seconds). Here lies the potential for the AML tank to blunder blindly into an extremely vulnerable position--a position, for example, from which it would have been well covered from the line-of-fire of the opponent tank at its former "present position" but would be totally exposed to the line-of-fire of the opponent tank in its now "present position." To avoid such pitfalls, provision is made at a higher level of priority for evaluating and accepting or rejecting each next impending move along the currently projected path pending selection of a new candidate position and determination of the updated path.

At this higher or second level of processing priority, a yes or no determination is made as to whether or not the next grid square along the currently projected path (i.e., the grid square into which the AML tank is about to move) falls within the weapon range and line-of-fire of the now "present position" of the opponent tank. If not, the move is allowed and the information processing reverts to the lower or third priority level where the process of selecting a new, longer-range candidate position and path is continued

until completed or preempted by a higher priority demand. If the impending move would place the AML tank within the range and line-of-fire of the opponent tank, the move is disallowed and an abbreviated immediate action process initiated to determine the best action. This abbreviated process uses the same Valuated State Space as for the longer range candidate position selection but consideration is confined to only nine grid squares . . . the grid square encompassing the AML tank's present position and those eight grid squares immediately adjacent to it.

Information processing at the higher or second level of priority, in addition to detecting and avoiding the high vulnerability pitfalls described above, is concerned also with generating and receiving the regular interchange of position data between the AML computer and the display apparatus. At specified intervals, say every one-sixteenth of a second, the AML program receives the position and heading of the opponent tank(s) and transmits the position, heading and turret aspect of the AML tank. This flow of information provides the necessary input for smooth visual presentation of the AML tank simulation as well as the input necessary to update the Valuated State Space profiles for evaluating and selecting the movements and action of the AML tank.

At the highest or first priority level are those routines that handle the appropriate responsive action when AML tank is fired upon by the opponent tank (i.e., the trainee).

SECTION II - VALUATED STATE SPACE

1. PURPOSE OF TANK COMMANDER IN COMBAT ENVIRONMENT

1.1 (10) Own Survival

1.1.1 (3) Avoid Detection

1.1.1.1 (10) Mask From Immediate Threat

10	4	2	0
Completely concealed	Mostly concealed	Partially concealed	Continually visible

1.1.1.2 (7) Mask From Air Detection

10	4	2	0
Completely concealed from overhead detection	Intermit-tently exposed to overhead detection	Intermit-tently con-cealed from overhead detection	Continually exposed to overhead detection

1.1.1.3 (3) Mask From Ground Detection

10	5	2	0
Masked from 3 quadrants or greater	Masked from 2 to 3 quadrants	Masked from 1 to 2 quadrants	Exposed to detection from all quadrants

1.1.2 (10) Avoid Damage From Enemy Weapons

1.1.2.1 (10) Avoid Damage From Immediate Threat Weapons

1.1.2.1.1 (10) Avoid Damage by Terrain Cover

10	7	3	0
Completely covered by terrain	Excellent terrain cover	Partial terrain cover	Exposed to line of fire-no terrain cover

1.1.2.1.2 (3) Avoid Effective Range of Weapons

10	5	2	0
Outside maximum range	At maximum range - armor front	At maximum range - vulnerable front	Within effective lethal range

1.1.2.2 (5) Avoid Damage From Air-to-Surface Weapons

10	5	2	0
Overhead cover completely blocks line-of-fire from aircraft	Intermittent breaks in overhead cover	Intermittent overhead cover	No overhead cover

1.1.2.3 (3) Avoid Damage From Surface-to- Surface Weapons

10	7	3	0
Terrain cover from 3 or more quadrants	Terrain cover from 2 to 3 quadrants	Terrain cover from 1 to 2 quadrants	No terrain cover

1.1.3 (5) Avoid Exposure to Enemy Acquisition and Tracking

1.1.3.1 (10) Avoid Presenting Silhouetted Target

10	7	5	0
Excellent background cover ~270°	Good background cover ~180°	Not silhouetted	Silhouetted (at crest of hill or ridge)

1.1.3.2 (7) Maintain Proximity to Cover

1.1.3.2.1 (10) Maintain Proximity to Air Cover

10	5	2	0
Cover within grid square	Cover within an adjacent grid square	Cover within 2nd grid square away	No cover within two grid squares away

1.1.3.2.2 (5) Maintain Proximity to Ground Cover

10	5	2	0
Cover within grid square	Cover within an adjacent grid square	Cover within 2nd grid square away	No cover within two grid squares away

1.1.4 (3) Maintain Maximum Maneuverability

1.1.4.1 (10) Avoid Steep (or Poorly Trafficable) Terrain

10	5	2	0
Level terrain with good surface	Sloping terrain reduced speed	Steep terrain barely trafficable	Impassable

1.1.4.2 (5) Avoid Heavily Forrested or Built-Up Areas

10	5	2	0
Unrestricted	Intermittent restrictions	Frequent restrictions	Heavily forrested built-up

1.2 (5) Enemy Destruction

1.2.1 (3) Maintain Visual Surveillance

10	7	5	0
Continuously in sight	Occassionally obscured	Intermit-tently in sight	Not in line of sight

1.2.2 (10) Gain or Hold Advantageous Firing Position

1.2.2.1 (10) Maintain Enemy Within Range of Weapon

10	7	5	0
Within effective range	At maximum range - vulnerable aim point	At maximum range - armor aim point	Not within range or arc of fire

1.2.2.2 (5) Maintain Favorable Relative Position

10	7	3	0
Line of fire to exposed vulnerable aim point	Line of fire to partially exposed vulnerable aim point	Line of fire to armor front, or hull down	out of range or arc of fire

1.2.3 (3) Gain or Hold Favorable Relative Exposure

1.2.3.1 (10) Cover From Air-to-Surface Weapons

10	4	2	0
AML in cover - enemy exposed	Both in cover	Both exposed	Enemy in cover - AML exposed

1.2.3.2 (7) Cover From Ground Fire/Shrapnel

10	4	2	0
AML in cover - enemy exposed	Both in cover	Both exposed	Enemy in cover - AML exposed

SECTION III - PROGRAM FLOW

In order to discuss the overall program flow,* certain assumptions are made concerning the hardware to be used:

1. Separate microprocessors for the AML program (AMP) and the display program (DSP).
2. Clock interrupt set to interrupt the AMP, say, every one-sixteenth of a second.
3. Communication channels between the AMP computer and the DSP computer with the DSP computer as master and the AMP computer as slave.
4. Possibly hardware interrupt upon trainee tank firing although this could be handled via messages from the DSP computer to the AMP computer.

Based on the previous assumptions, AMP flow can now be discussed. The one-sixteenth second clock interrupt is required because a message must be sent approximately each one-sixteenth of a second upon request by the DSP computer. This message consisting of the AML tank(s) position, heading and relative bearing of the turret with respect to the AML tank(s) fore and aft axis. Similarly, a message must be received by the AMP concerning trainee's tank. These messages are for display and data processing purposes, respectively.

---

* This section may best be followed by referencing the flow diagrams given in Section VI.

As previously discussed, there are three levels of priority:

1. AML tank fired upon.
2. a. AML tank proceeding on the path determined by the Priority Three program.
   b. Updating present AML tank's position and message composition.
   c. Decoding DSP computer message.
   d. In the event that an impending move by the AML tank would make it too vulnerable, determine immediate action regardless of the path determined by the Priority Three program.
   e. Fire on enemy tank(s) if able and desirable.
3. Select desired position and determine path to the desired position.

For the purpose of discussion assume the AMP is operating at Priority Three and a clock interrupt occurs. The return address is saved, of course, and control is transferred to Priority Two where the AML tank position is updated and a message is formatted for transmittal to the DSP computer. A message from the DSP computer is decoded and the trainee tank's position is updated. If informed by the DSP message that the AML tank has been fired upon or upon determining too great an AML tank vulnerability, control transfers to the Immediate Action section of Priority Two and the previous plan of proceeding along the projected path is abandoned.

If, however, neither vulnerability exists and the tanks are still in the same grid squares, control transfers back to the Priority Three level at the place in the program where the clock interrupt took control. On the other hand, if the trainee tank has moved to a new grid square or the AML tank has moved off the previously determined path, control is transferred to the beginning of the Priority Three level program. Note that this requires that the Priority Three level program be re-entrant, that is, this routine may be started at the beginning at any time and the program will execute correctly regardless as to whether the program was left previously in a completed or partially completed state. If a path has been determined and the trainee tank is still in the same grid square used in determining the path, then there is no point in entering the Priority Three program until the trainee tank has moved into a different grid square. This situation is expected to occur frequently.

The Immediate Action portion of the program, as mentioned before, makes the determination of initiating or returning fire, and either remaining in the present grid square or moving to one of the eight adjoining squares. This action is determined through use of the same Valuated State Space used throughout the program.

However, the question as to whether or not the tanks are in range should be calculated from the actual positions as opposed to using the information stored in the associated

fixed record,* this because the stored information was computed on the average distance between grid squares. If the trainee tank has traversed from its former position into a different grid square or the AML tank has moved off the previously determined path, the Priority Three level program subsequently should be re-entered at the beginning. If not, there is no use in its being entered. Therefore, after the Immediate Action processing has occurred, subsequently a test is made as to whether or not the trainee tank has moved into a different grid square or the AML tank has moved off the previously determined path and, if so, control transfers to the beginning of the Priority Three program. Otherwise, Priority Three is re-entered via interrupt return, if necessary.

Note provision for an idling loop, this to permit a measure of processor utilization.

* See next section for record description.

SECTION IV - RECORD STRUCTURE

Each of the grid points (areas) has an associated fixed record which contains information descriptive of that area and certain precomputed values describing the relation of that area to each of the other areas, for example, can a tank in the one area be seen from a tank in the other area.

Excluding any header information, the record consists of three sections.

Section One consists of five words containing the identification of up to ten potential candidate positions as desireable positions whenever the opponent tank is in the area associated with the record.

Section Two consists of one word, say, eight two bit fields concerning:

Concealment From Air Detection-General

Two Bit Field	
11	Not Detectable - Overhead cover completely obscures/blocks direct line-of-sight from aircraft
10	Detection Possible - Intermittently exposed to direct line-of-sight from aircraft due to occasional breaks in overhead cover
01	Detection Probable - Some overhead cover but usually exposed to direct line-of-sight from aircraft

00 Detection Almost Certain - In exposed, open field of view from the air

Concealment From Ground Detection - General

Two Bit Field	
11	Excellent Concealment - Terrain features mask detection from three or more quadrants
10	Good Concealment - Terrain features mask detection from two to three quadrants
01	Fair Concealment - Terrain features mask detection from one to two quadrants
00	No Concealment - Exposed to detection from all quadrants

Target Vulnerability - Acquisition

Two Bit Field	
11	Excellent Background Cover (~270°)
10	Good Background Cover (~180°)
01	Not Silhouetted (~90°)
00	Silhouetted

Target Vulnerability - Tracking (Air)

(Duration of vulnerability based on proximity to air cover.)

Two Bit Field	
11	Cover Within Grid Square
10	Cover Within An Adjacement Grid Square

01	Cover Within Second Grid Square Away
00	No Cover Within Two Grid Squares Away

Target Vulnerability - Tracking (Surface)

(Duration of vulnerability based on proximity to ground cover.)

Two Bit Field	
11	Cover Within Grid Square
10	Cover Within An Adjacent Grid Square
01	Cover Within Second Grid Square Away
00	No Cover Within Two Grid Squares Away

Trafficability

Two Bit Field	
11	Passable at Cross-country Speed (30 mph)
10	Passable at Reduced Speed (15 mph)
01	Passable at Very Slow Speed (5 mph)
00	Impassable

Terrain Cover From Surface Weapons - General

Two Bit Field	
11	Excellent Cover (~270°)
10	Good Terrain Cover (~180°)
01	Modest Terrain Cover (~90°)
00	No Terrain Cover

Terrain Cover From Air Weapons - General

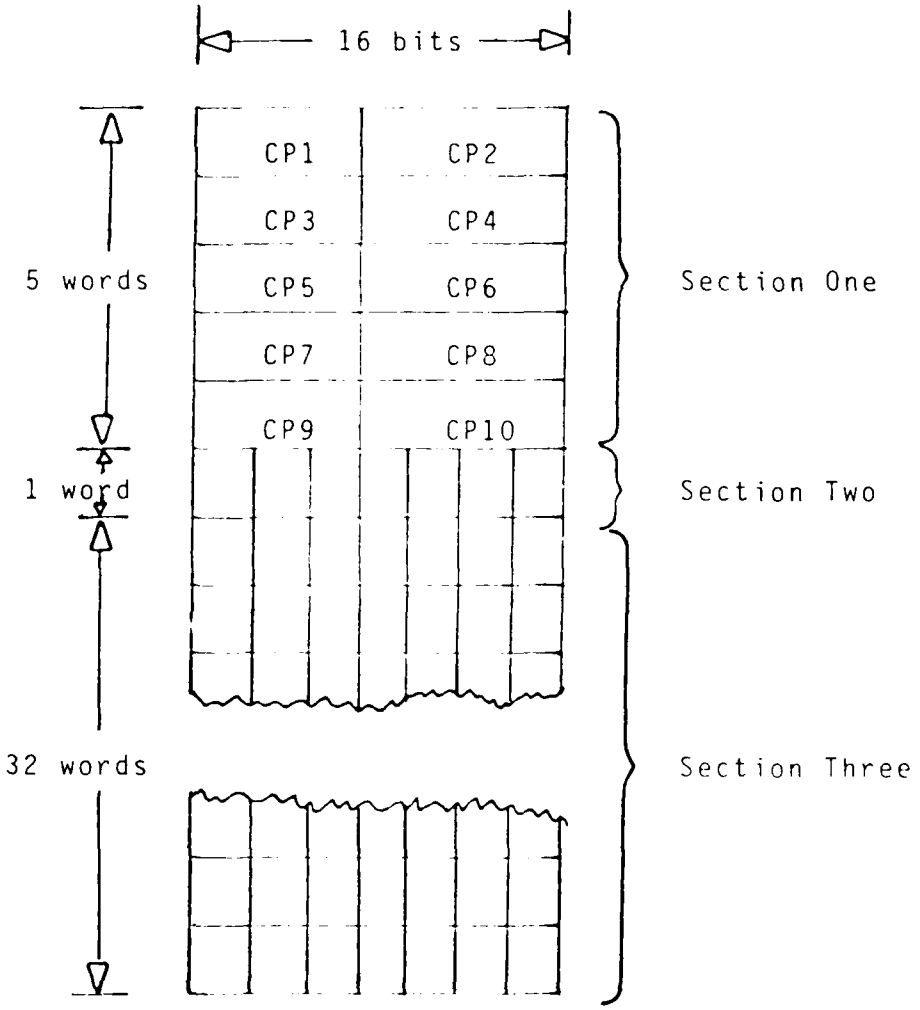
Two Bit Field	
11	Complete Overhead Cover
10	Occasional Breaks in Overhead Cover
01	Scattered Overhead Cover
00	Completely Exposed - No Overhead Cover

Section Three consists of 32 words with 256 two bit fields concerning:

Detectability From Each of the Other 256 Grid Areas

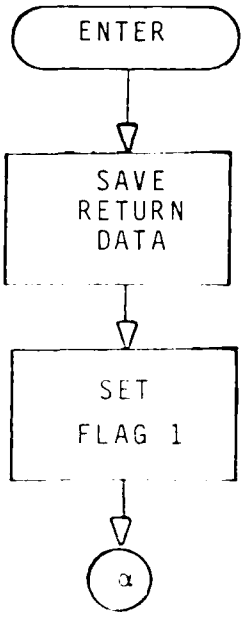
Two Bit Field	
11	Not Detectable - Not in line-of-sight
10	Detection Possible - In line-of-sight but mostly concealed by features of terrain
01	Detection Probable - In line-of-sight and partially concealed by features of terrain
00	Detection Almost Certain - In line-of-sight and in open-field of view

RECORD FORMAT

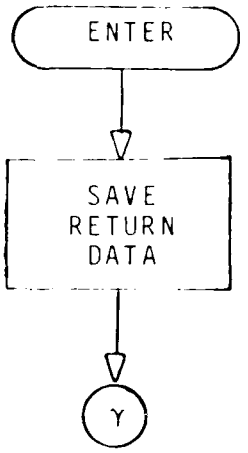


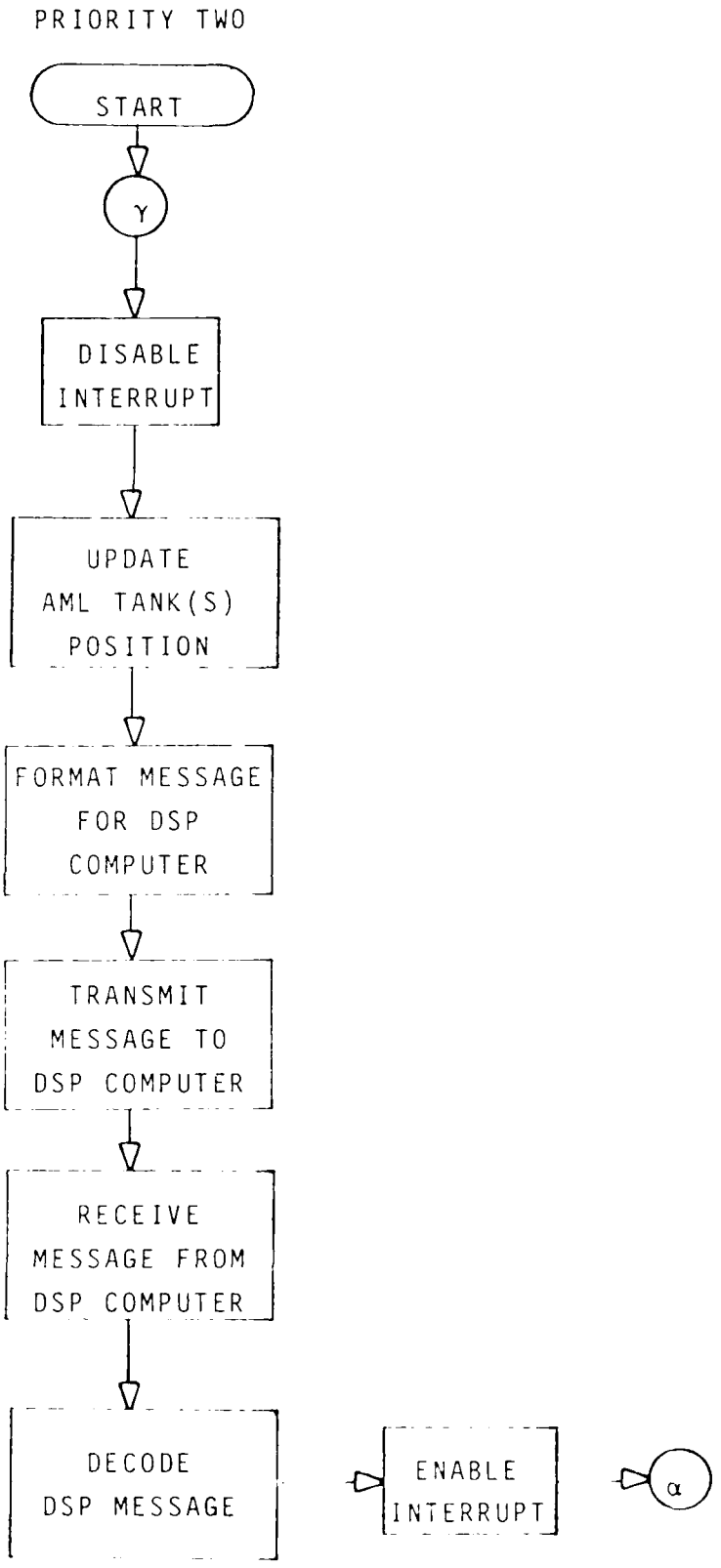
SECTION VI - FLOW DIAGRAMS

PRIORITY ONE
(IF HARDWARE FIRING INTERRUPT)

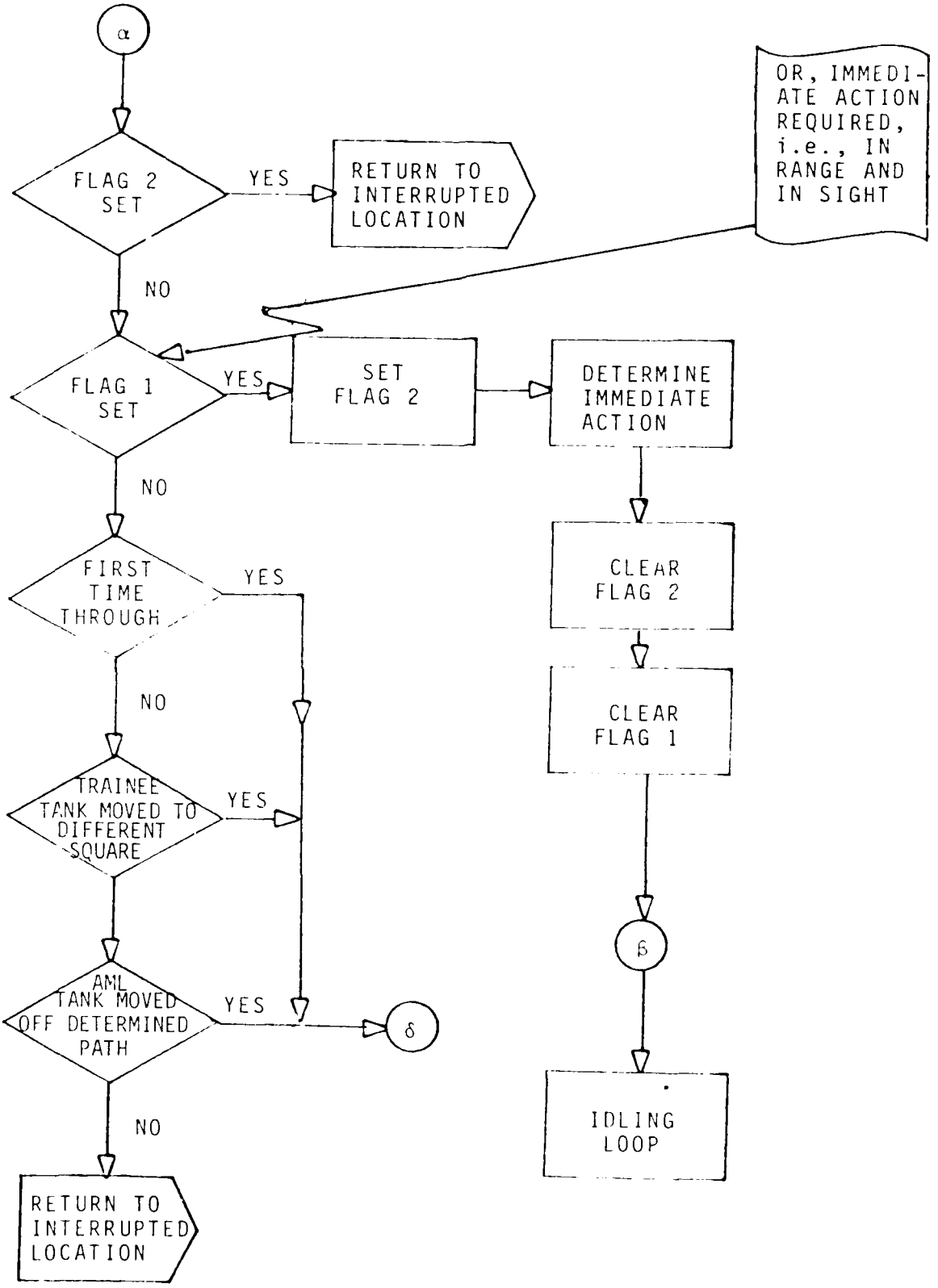


CLOCK INTERRUPT

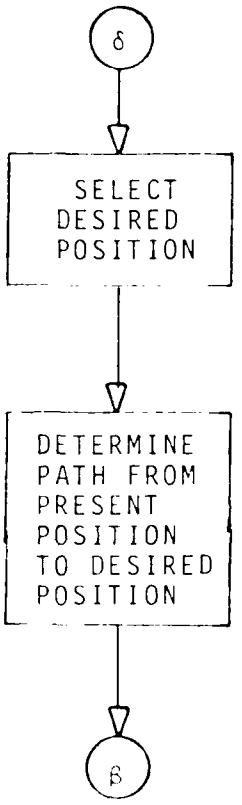



PRIORITY TWO
START
γ
DISABLE INTERRUPT
UPDATE AML TANK(S) POSITION
FORMAT MESSAGE FOR DSP COMPUTER
TRANSMIT MESSAGE TO DSP COMPUTER
RECEIVE MESSAGE FROM DSP COMPUTER
DECODE DSP MESSAGE
ENABLE INTERRUPT
α

PRIORITY TWO (CONTINUED)



PRIORITY THREE



SECTION VI - SUMMARY

The objective of this research was the development of algorithms and preliminary specifications for a computer program to intelligently interactively control tank(s) during simulated tank warfare. The technical problems being that, regardless of the initial situation or subsequent development of the interaction between the human trainee and the simulated tank, the simulated tank would not maneuver stupidly, rather, it would maneuver in such a way as to be a worthy opponent and avoid inadvertent high risk maneuvers. Four basic problems were identified and solutions found: first, a criteria to measure the worth of a position; second, candidate (desired or advantageous) positions; third, determine path from current position to the desired position; fourth, traverse the path, guard against inadvertent (stupid) action and, if such action be eminent, take correcting offensive or defensive action, as appropriate.

The scope of the contract did not permit programming and exercising the algorithms developed. However, based on previous experience with application of the Adaptive Maneuvering Logic, it is thought a worthy opponent can be implemented using the algorithms developed.

The emphasis of this effort was primarily concerned with the decisions required of a platoon tank commander and

the maneuvering of his tank; however, the analysis and the structure of the program was developed with the thought that it be enlargeable to maneuvering a platoon of four tanks.

Further research is required to address the maneuvering and coordination of a platoon of four tanks.

APPENDIX A

FIRST PROGRESS REPORT

CONTRACT NO. MDA903-81-C-0509

DECISION SCIENCE, INC.
4901 MORENA BOULEVARD
SAN DIEGO, CALIFORNIA
92117 (714) 273 2922

DSI-81-413-1

THE ADAPTIVE MANEUVERING LOGIC IN TANK WARFARE SIMULATION

PROGRESS REPORT

Contract No. MDA903-81-C-0509

Submitted To:

Major Jack Thorpe
USAF, DARPA/CTD
Cybernetics Technology Division
1400 Wilson Boulevard
Arlington, VA 22209

January 12, 1982

INTRODUCTION

Combat training is effective only if the simulated enemy does not follow a canned scenario. Clearly, the gaming must be interactive. In the real world there is always a presumption that an enemy's moves reflect both his mission and your actions.

The Adaptive Maneuvering Logic (AML) is a unique means for simulating intelligently interactive gaming. Here the purpose of each of the players is defined in a concise manner. Reference is made to possible moves (alternative commitment of resources). Each such move is translated into a new overall worth, taking into account both own purpose and the purpose of the other player(s). A comparison of these moves then allows selection of the best move at that point in time. This fast time evaluative process proceeds while the game is played in real time.

The AML can be used to drive the enemy force and/or used to score the human operator (by playing in parallel with him). Here any significant differences of "opinion" are accompanied by a concise rationale.

DISCUSSION

The primary purpose of all animate beings is survival. The purpose of the tank commander concerns preserving his physical embodiment and his self-image. In turn, each of these aspects of survival can be particularized. For example, physical survival includes concern for killing the enemy and/or avoiding being killed. In turn, killing the enemy can be achieved through calling for support, shooting first, using entrapment, and so forth. Each of these aspects can be further particularized. Table 1 (page 14) indicates these dimentions of purpose in hierarchic format. The Dewey Decimal notation is used to facilitate reference to individual branches within such a tree structure.

Note that the relative importance of these dimensions of purpose depends strongly on the circumstance. Lower order animals are primarily concerned with physical survival, the concept of the self being only partially developed. In contrast, the martyr will willingly goes to his death to preserve what he stands for. Here psychological self-preservation is far more important that preserving the physical self.

In some combat situations killing the enemy is all important and it is acceptable to be killed, as witness Kamikaze attacks. In other situations, killing the enemy may be far less important than preserving one's self . . . to live and fight another day. Indeed, the relative importance weights are peculiar to the mission, that particular situation, and that time.

Each of these parameters of purpose must be explicated at successively lower levels until they are made operational. That is to say, each lowest level parameter must be measurable in terms of mutually exclusive class intervals that define those differences that make a difference in degrees of achievement.

These class intervals must exhaustively span the range, each being attributed some value for that degree of achievement. In general, more important parameters are referenced with greater specificity. If air support is very important, then the degree of air support achieved should be indicated in some precise sense (up to 7 $\pm$ 2 categories). If, on the other hand, air support is relatively unimportant then the degree of achievement can be binary (either there is air support or not). Here again, the nature of the class intervals and their values depend upon the particular circumstance (the assigned mission and even the developing situation).

Clearly, such a complex statement of mission goes far beyond the usual orders or assignments. Command such as "Take that hill" are too ambiguous. Meaningful interpretation requires explication of what that command really means to the decision-maker. Here the endeavor is to explicate that rationale so that it can be readily referenced. Table 1A (page 17) represents a refinement of Table 1 particularized for the selected combat scenario of the Central European Front (Fulda Gap) and further explicated to reflect the purpose and operational option of the individual tank commanders.

Purpose requires a statement of a hierarchic valuated state space and appropriate normalizing function. It is common to use the weighted arithmetic mean as the normalizing function because of its simplicity. Here the overall worth of the present situation (a profile of class intervals across the measurable parameters) is computed by summing the weighted contribution factors. Some situations might reference critical parameters. Here it is more appropriate to use an alternative normalizing function, such as the weighted geometric mean, wherein complete failure on a single parameter nullifies the overall degree of success. In realistic situations, the normalization is a composite of different functions that reflect the nature of each of the parameters. This normalizing function tends to remain invariant during the combat.

But the game is not well defined unless the purpose of the opposing force is also taken into account. In the two player game this requires expressing the enemy's mission in equally concise terms. In real world situations it is necessary to infer this purpose based on intelligence, the commitment of enemy forces, their demonstrated capability, and the willingness to risk facing a more committed enemy than was presupposed in making the decison to initiate the combat engagement.

As defined above, the hierarchic valuated state space provides a discrete scale of overall achievement ranging from catastrophe to utopia (from zero to 100%). The number of class intervals on the scale is the product of the number of class intervals on each of the operational parameters. This scale represents a linear array of overall degrees of achievement for each of the situations considered to be significantly different in the light of the purpose. A similar definition of the enemy's purpose can be used to define the joint state space that defines the game, reference Figure 1. Here each cell corresponds with a significantly different situation from either or both player's points of view. Once the game is defined by this two-space, the marginal payoffs become meaningless. Each cell in the joint state space designates a payoff to each of the players for that situation. By convention, the purpose of the primary player of concern is expressed across the top, while the purpose of the other player is expressed on the left side of the matrix. A diagonal in each cell separates the two payoff's, the upper right payoff being to the primary player, while the lower left payoff is to the other player.

The nature of the game is expressed by the joint payoff function. If the payoff is the same over an entire column for the primary player, then he is unconcerned about the degree of achievement of the player while in that state. A competitive attitude is expressed if the descending column increases in payoff.

If the column decreases, the game is cooperative in this domain. A similar logic can be used to exmiane the rows from the prespective of the other player. Note that, in general, games may be cooperative, competitive or ambivalent to different degrees and in different states. Even a mutually cooperative game may be asymetrical in that the degree of cooperation is different for each of the players.

The line items of the purpose of each player dictate the questions that must be answered concerning the present and projected situation. The present degree of achievement across the measurable parameters of both players' purpose is translated into an overall payoff. At this point, it is pertinent to examine the prospective moves in the game (combinations of the allocable resources). If the decision-maker is the tank commander, then these resources are those aspects he can commit in the present situation. For example, he can cause his tank (and perhaps other tanks) to move toward, away from, or circumvent an enemy position or selected feature of terrain. He can fire his main or auxilliary weapon, button up, or choose to remain open; and, in some situations, he may choose to accept fuel, ammunition, food, replacement crews, and so forth.

Table 2 (page 26) indicates these resources during typical combat. The class intervals being degrees of commitment of individual resources. Note that here there is no need for relative weights and values. The listing is merely a table of what can be committed. Each profile across the class intervals is a prospective move. The task is to evaluate the relative worth of these, given the purpose and the present situation.

Each considered move must be translated into a corresponding trajectory in the joint state space. Note that there is no continuity in this state space. It is merely an array of all possible situations and an indication of their worth to each

of the players. Moving forward at a certain speed changes the profile of degree of achievement of purpose depending on the particular terrain. In some cases forward movement might be effective for hiding; in others, it might make the tank more vulnerable. Here the dynamics of the physical situation must be referenced. This includes the topography, weather, and motion of all involved moving platforms.

Intelligently interactive gaming presumes that the enemy is purposive and also exercises a similar logical capability. The trajectory in the joint state space must therefore take into account the commitment of resources by the enemy (his countermoves). Each of the alternative initial moves calls for a probabilistic tree of countermoves. It is common to face uncertainty concerning enemy moves in response to the initial moves and at further steps in the tree. Each move is defined as an observable event, the time required for this event to take place, and the probability of that event as estimated. A determinnistic strategy is defined as a tree of moves and countermoves wherein the decision as to how we are to respond is deterministic for each of the other player countermoves. In some situations there may be uncertainty in one's own strategy. Here the alternative moves are defined as a commitment (an event), the required time for that event, and its probability. In general, probabilistic strategies are less valuable than deterministic strategies.

Each of the alternative initial moves then corresponds with a trajectory in the joint state space and an associated payoff function over further time. The task at hand is to compare these functions in order to determine the best move of those considered. (It is recognized that an exhaustive search of all possible moves is unrealistic and indeed unwarranted. Heuristics are generally available to direct attention to those few alternative moves that are most worthy of attention.)

A simplistic view references the tree of moves and countermoves growing out of each alternative initial move in terms of the expected payoff at the end of each branch (scenario). These payoffs can be aggregated into a measure of the overall worth of the initial move, taking into account whatever probabilistic and temporal discounting may be involved. The alternative moves can then be ranked. A more sophisticated view, however, recognizes the fact that each player holds a metapurpose in addition to his purpose so that each alternative move is evaluated in terms of payoff and the problems posed as costs associated with that payoff. This subject can be treated in greater detail, but is beyond the scope of this investigation. The task at hand is to demonstrate the AML in relatively simple situations of tank warfare where [illegible] presumptions are made concerning purpose, resources, and the [illegible] environment.

This requires recognition of the [illegible] aspect of purpose for each of the players. [illegible] trajectory. The question is "Is that acceptable?" [illegible] recognize the stringency of his goal seeking. Must [illegible] by a certain time, or is he satisfied with achieving [illegible] certain date? In real situations the metapurpose is often less stringent [illegible] "Let's not go downhill" and, in some cases, any degree of success is considered acceptable.

Problems are defined by the recognized differences between the expected and desired degrees of success. In general, more stringent metapurpose generates a larger array of problems, each being defined by the expected date of onset, duration, and degree of severity. Less striving generally produces fewer problems and, in the limit, the lackadaisical player has no problems. Purposive play requires recognition of both the purpose and the metapurpose of each of the players.

The preceeding has been an overview of a general methodology for approaching the problem of tank warfare simulation. In the next section this methodology is further elaborated in greater detail.

PRELIMINARY SOFTWARE SPECIFICATIONS FOR TANK WARFARE

I Introduction

The problem of designing an Adaptive Maneuvering Logic for controlling the actions of one or more tanks in a battlefield simulation is a considerably larger task than controlling the actions of one or more fighter planes in air-to-air-combat simulation. Among the factors complicating the task are:

1. Terrain Features: Certain positions on the battlefield are inherently advantageous due to cover, visability and so forth. These shall be called candidate positions.
2. Number and Length of Trial Maneuver Paths: Movements of an airplane are limited mainly by its maneuvering capabilities, its altitude capabilities, and the earth's surface. And for most maneuvering the latter two liminations do not apply. Tactical decisions are made rapidly and simulation flight paths can be constructed from elemental tactical moves. But with tanks, tactical moves may be based on movement over a span of minutes, making it necessary to consider and evaluate many more alternate paths. Thus, unlike airplanes, tank decision control cannot be practically accomplished by elemental decision maneuvers.
3. Multiple Variables: More parameters are involved in the tank simulation as compared with air-to-air-combat simulation -- concealment, fog, rain, tank noise, armor thickness (as it influences maneuvering), buttoned up or not, minefields, artillery support, and so forth.

This is not to say, however, that it is an undoable task. In subsequent sections, preliminary specifications are laid down for a computer program which will accomplish the required decision control in a large class of situations. The method involves a judicious combination of valuated state space techniques and the Adaptive Maneuvering Logic -- two flexible approaches previously employed by Decision Science, Inc. with considerable success. A model of the overall mission or purpose of the battlefield operations is provided by a valuated state space while the details of complex tactical movements are carried out by the Adaptive Maneuvering Logic.

II The Grid

In the air simulations, for all practical purposes, relative positions of the aircraft is the predominant consideration (except, perhaps, for the direction of the ground and the sun). But in the tank simulation, each land area has terrain features that in general cannot be ignored -- buildings, roads, hills, trees, boulders, rocks, sand, marsh, and so forth. In order to keep track of all these, the battlefield region is partitioned into a grid of squares. The length of a side of any one of the these grid squares can be set, say, between 10 and 50 meters. To each grid square is associated a set of descriptive values -- hilly or flat, rocky or smooth, trees and other cover or open, elevation, passable or impassable, and so forth. Exact sizes and positions of buildings and possibly other features are recorded by xy-coordinates. In addition, at any time the positions of all friendly and enemy tanks can be defined in terms of either these grid squares or the coordinates.

A Utility Function, consisting of a composite of attackability and survivability, is computed for a tank relative to a second (enemy) tank using the data associated with the position squares of the tanks and other data. The

Utility of an action by a tank [illegible] survivability and attackability. [illegible]

III Valuated State Space of the Mission

The mission of a tank (or group of tanks) greatly determines the Utility of a given tactical maneuver. While a tactical retreat in the face of a confrontation might be appropriate in one mission it could be inappropriate in another, more aggressive mission. The relative importances of survivability and attackability must be defined and determined in terms of the particular overall mission governing the tank engagement. Once the relative importances of survivability and attackability have been determined, the Utility of occupying a position in a grid square can be assessed.

A preliminary valuated state space quantifying various tank missions is given in Table IA. Both the survivability and attackability of a tank in a given grid square depend upon various parameters -- range to enemy tanks, available cover, and so forth. Together, survivability and attackability determine the overall Utility of a position.

IV Optimal Tank Moves and Adaptive Maneuvering Logic

Having laid down the battlefield context (the grid, the xy-coordinates, and the associated terrain data) together with determining the Utility of a given tank position relative to a second (enemy) tank in view of the mission and in terms of survivability and attackability, it remains to lay out a method for using the Adaptive Maneuvering Logic to find advantageous paths to advantageous grid squares and to select a path for action. A path is advantageous if all of its grid squares have high survivability, perhaps with high attackability, this, of course, dependending upon the mission.

As conceived, tank maneuvers (tactics) are determined as follows: an initial position is given for the friendly and enemy tanks. Given the initial position of the enemy tanks, each grid square in the field (or as many nearby the friendly tank as real time permits) is evaluated as to its Utility (survivability and attackability), relative to the enemy tank at its present position under the assumption the friendly tank is instantaneously transported to those grid squares. In this way, potentially advantageous positions for the friendly tank can be identified.

Once several candidate squares have been identified, they are ordered according to their Utility to the friendly tank. Some of these candidate squares can be eliminated from immediate consideration: if the enemy tank can move to a square which makes the friendly tank's Utility at the candidate square less than it was at the initial positions of the two tanks, that candidate square is temporarily rejected and a new candidate square is evaluated. In this way a new ordering for the candidate squares is obtained depending upon the possible responses of the enemy tank.

Define:

U_F as the friendly tank's Utility at its present positon and the enemy tank at its present position.

U_F' as the friendly tank's Utility at a candidate position and the enemy at its present position.

U_F'' as the friendly tank's Utility at a candidate position and the enemy tank at some position "nearby its present position." For each candidate square determine the smallest U_F'' for all "nearby squares" that the enemy tank may occupy. Order the candidate square by maximum change in Utility, $U_F'' - U_F$. Once several possible advantageous moves are identified, there remains the problem of maneuvering the tank.

A choice is made based upon whether or not a satisfactory route can be found to the candidate position, the time required to attain that position, and a function of the collected Utilities of the relative positions of the tanks as the position is being attained. Note that the survivability of a tank traversing a path over grid squares is a product of the survivability of the tank at each square it traverses. The survivability of a tank at a given grid square is the probability of its not being destroyed while occupying this grid square for one unit of time. If the tank moves through even one square which has low survivability, then the path also has low survivability no matter how survivable the tank is at other squares on the path.

If a safe path (one with a certain minimal survivability) is found leading to a highly advantageous grid square, the friendly tank begins to traverse that safe path. But if there are no safe paths to the highly advantageous square, that grid square is discarded from immediate consideration; and a different highly advantageous square is evaluated for path survivability. If there are no safe paths to any highly advantageous squares, then a deeper assessment of all candidate squares is begun. It may be that a temporarily rejected candidate square is in fact actually quite advantageous because there is no safe way for an enemy tank to move to a square that would make the candidate square disadvantageous. To determine this the paired positions of the friendy and enemy tanks are assessed with respect to the survivability of each at each grid square along their respective paths -- the friendly tank as it travels to the candidate square and the enemy tank on its way to a square that makes the candidate disadvantageous. If the overall survivability of the path of the enemy tank is low enough (determined by the mission valuated state space), then there is little danger that the enemy tank will attempt to traverse this path and occupy the disadvantaging square. If similar calculations rule out all other paths

to the disadvantaging square, then this square is deleted from the list of disadvantaging squares for the given candidate square. If all possible disadvantaging squares for this given candidate square are found to be unsafe for the enemy tank to approach, then the candidate square is qualifiedly advantageous, since no enemy tank can safely pass to any disadvantaging square. Action results immediately -- the friendly tank begins traversing the path to the qualifiedly advantageous grid square. If no highly or qualifiedly advantageous candidate square is available, then the tank may remain immobile or move to some safe square, depending upon the mission.

In evaluating any given potential move by the friendly tank into an adjacent position (including the present position) it is necessary to calculate the overall utility (in terms of survivability and attackability) of this potential friendly tank move coupled with a move by any enemy tank to one of its adjacent positions (including its present position). If this overall utility is unacceptably low, then this position is avoided by the friendly tank.

Throughout these calculations, a monitor routine checks the survivability and attackability of the friendly tank during the present instant. If ever the survivability falls below a certain minimum (depending on the mission) or if attackability becomes high, then appropriate action is taken.

The Adaptive Maneuvering Logic is of course interactive. A completely new evaluation is initiated as real time permits.

TABLE 1

THE ADAPTIVE MANEUVERING LOGIC IN TANK WARFARE

(A GENERALIZED "FIRST-CUT")

1. Survival

1.1 (10) Physical Survival

1.1.1 (5) Kill the Enemy

1.1.1.1 (8) Call for Support

1.1.1.1.1 (10) By Artillary (*)

1.1.1.1.2 (7) By Air

1.1.1.1.3 (5) By Infantry

1.1.1.2 (10) Shoot First

1.1.1.2.1 (1) Move to Contact

1.1.1.2.2 (6) Ambush

1.1.1.2.3 (0) Buckshot

1.1.1.2.3 (10) Alert

1.1.1.3 (3) Entrap

1.1.1.4 (1) Entice into Danger

1.1.1.5 (7) Immobolize

1.1.2 (10) Avoid Being Killed

1.1.2.1. (7) Avoid Detection (Hide)

1.1.2.1.1 (10) Mask

1.1.2.1.1.1 (10) Using Terrain

1.1.2.1.1.2 (7) Using Camouflage

1.1.2.1.1.3 (2) Using Smoke

(*) NOTE: Appropriate Class Intervals follow the lowest parameter in each case.

1.1.2.1.2 (2) Diversion

1.1.2.1.2.1 (1) Using Smoke

1.1.2.1.2.2 (8) Using Support Forces

1.1.2.1.2.3 (0) Leaving Multiple Trails

1.1.2.1.2.4 (0) Using Decoys

1.1.2.2 (10) Avoid Enemy Weapons (Minimize Vulnerability)

1.1.2.2.1 (10) Shield by Terrain

1.1.2.2.2 (7) Neutralize Enemy Weapons

1.1.2.2.2.1 (1) Using ECM

1.1.2.2.2.2 (0) " Chemical Means

1.1.2.2.2.3 (5) " Smoke

1.1.2.2.2.4 (7) By Disrupting His Solution

1.1.2.2.2.5 (1) By Threat

1.1.2.2.2.5.1 (5) Using Buckshot

1.1.2.2.2.5.2 (2) " False Communications

1.1.3 (7) Repulse Attack

1.1.3.1 (8) Block Advance

1.1.3.1.1 (10) Through Physical Means

1.1.3.1.1.1 (5) Mines

1.1.3.1.1.2 (0) Chemical Warfare

1.1.3.1.1.3 (3) Tank Trap

1.1.3.1.2 (2) Psychological Fear

1.1.3.2 (10) Force Retreat

1.1.3.2.1 (10) Physical Exercise of Power

1.1.3.2.2 (3) Psychological Fear

TABLE 1 (CONTINUED)

1.2 (3) Psychological Survival (Self-preservation)

1.2.1 (10) Avoid Being A Coward

1.2.1.1 (5) Minimize Responsibility (Non-volunteer)

1.2.1.2 (10) Accept Assigned Risk

1.2.2 (8) Be Stalwart

1.2.2.1 (10) Demonstrate Competence

1.2.2.1.1 (10) Skill

1.2.2.1.2 (5) Rested Condition

1.2.2.1.3 (8) Ready Equipment

1.2.2.2 (8) Demonstrate Dedication

1.2.3 (2) Be a Hero

1.2.3.1 (10) By Being Clever

1.2.3.2 (1) Be Generating Charisma

1.2.3.3 (1) By Successful Risk Taking

TABLE 1A

THE ADAPTIVE MANEUVERING LOGIC IN TANK WARFARE (PARTICULARIZED FOR THE SELECTED EXAMPLE)

1. PURPOSE OF TANK COMMANDER IN COMBAT ENVIRONMENT

1.1 (10) OWN SURVIVAL

1.1.1. (3) AVOID DETECTION

1.1.1.1. (10) MASK FROM ENEMY VISUAL DETECTION

1.1.1.1.1. (10) BY USE OF TERRAIN

10	7	5	3	0
COMPLETELY CONCEALED	>75% CONCEALED	75% > 50% CONCEALED	50% > 25% CONCEALED	COMPLETELY EXPOSED

1.1.1.1.2. (2) BY USE OF SMOKE

10	5	0
EFFECTIVE AND PERSISTING	EFFECTIVE BUT RAPIDLY DISSIPATING	INEFFECTIVE

1.1.1.1.3. (5) BY USE OF CAMOUFLAGE

10	7	3	0
EFFECTIVE TO WITHIN WEAPON RANGE	EFFECTIVE AT LONG-RANGE (BEYOND WEAPON RANGE)	IMPEDES CASUAL DETECTION	NOT EFFECTIVE

1.1.1.2. (3) MASK FROM ENEMY ELECTROMAGNETIC DETECTION

1.1.1.2.1 (10) BY COMMUNICATIONS-ELECTRONICS SILENCE

10	3	0
COMPLETELY SILENT	INTERMITTENT RADIATION	UNRESTRICTED EMISSIONS

1.1.1.2.2. (3) BY JAMMING

10	7	5	0
FULL SPECTRUM JAMMED	SELECTED FREQUENCY JAMMED	SELECTED FREQUENCY INTERMITTENT	NO JAMMING

1.1.1.2.3. (5) BY MERGING WITH TERRAIN

10	3	0
TOTAL RADAR MASKING POSSIBLE	PARTIAL RADAR MASKING POSSIBLE	NO TERRAIN MASKING FEATURE AVAILABLE

1.1.1.3. (5) MASK FROM ENEMY ACCOUSTIC DETECTION

1.1.1.3.1 (10) BY ACCOUSTIC SILENCE

10	8	4	0
COMPLETELY SILENT	SILENT TO AUDIBLE DETECTION BEYOND FIELD OF VISIBILITY	SILENT IN ACCOUSTICALLY IDENTIFIABLE EMISSIONS	NOISY AND ACCOUSTICALLY IDENTIFIABLE EMISSIONS

1.1.1.3.2. (5) BY ACCOUSTIC JAMMING

10	3	0
SELF-NOISE OBLITERATED	SELF-NOISE OBSCURED	NO JAMMING

1.1.1.4. (2) DIVERT ENEMY ATTENTION ELSEWHERE

1.1.1.4.1. (10) BY SMOKE

10	5	0
EXCLUSIONARY DIVERSION	MOMENTARY DIVERSION	ALERT ENEMY TO PRESENCE

1.1.1.4.2. (5) BY COMMUNICATIONS DECEPTION

10	5	0
LASTING DIVERSION	MOMENTARY DIVERSION	ALERT ENEMY TO PRESENCE

1.1.1.4.3. (2) BY ACCOUSTIC DECEPTION/DECOY

10	5	0
EXCLUSIONARY DIVERSION	MOMENTARY DIVERSION	ALERT ENEMY TO PRESENCE

1.1.1.4.4. (2) BY ELECTROMAGNETIC DECEPTION/DECOY

10	5	0
EXCLUSIONARY DIVERSION	MOMENTARY DIVERSION	ALERT ENEMY TO PRESENCE

1.1.2. (10) AVOID DAMAGE FROM ENEMY WEAPONS

1.1.2.1. (10) AVOID DAMAGE FROM SURFACE-TO-SURFACE WEAPONS

1.1.2.1.1. (10) SHIELD FROM EFFECTS OF WEAPONS

1.1.2.1.1.1. (10) BY USE OF TERRAIN FOR COVER

10	7	5	3	1	0
COMPLETELY COVERED	>75% COVERED	50 > 75% COVERED	20 > 50%	10 > 25%	COMPLETELY EXPOSED

1.1.2.1.1.2. (5) BY USE OF PROTECTIVE ARMOR

10	6	2	0
ARMOR FRONT TO ENEMY LINE OF FIRE	ARMOR FRONT TO > 50% ARC OF FIRE	GREATEST VULNERA-BILITY TO >50% ARC OF FIRE	GREATEST VULNERABILITY EXPOSED

1.1.2.1.1.3. (3) BY USE OF PERSONNEL PROTECTIVE MEASURES

1.1.2.1.1.3.1. (10) BY "BUTTONING-UP" TANK

10	0
BUTTONED-UP	HATCHES OPEN

1.1.2.1.1.3.2. (5) BY PROTECTIVE CLOTHING/GAS MASK

10	0
YES	NO

1.1.2.1.2. (5) IMPAIR ACCURACY OF WEAPON

1.1.2.1.2.1. (10) BY DISRUPTING GUIDANCE

1.1.2.1.2.1.1. (10) BY SUPPRESSIVE FIRE

10	0
GUIDANCE SOURCE WITHIN OWN WEAPON RANGE	GUIDANCE SOURCE BEYOND OWN WEAPON RANGE

1.1.2.1.2.1.2. (3) BY ELECTRONIC/IR COUNTERMEASURES/DECOY

10	0
WEAPON SUSCEPTIBLE TO COUNTERMEASURES	WEAPON NOT SUSCEPTIBLE

1.1.2.1.2.1.3. (5) BY "JINKING" MANEUVERS

10	0
EFFECTIVE	NOT EFFECTIVE

1.1.2.1.2.2. (5) BY OBSCURING 'POINT OF AIM'

1.1.2.1.2.2.1. (10) BY SMOKE

10	0
EFFECTIVE	NOT EFFECTIVE

1.1.2.1.2.2.2. (5) BY TERRAIN

10	0
EFFECTIVE	NOT EFFECTIVE

1.1.2.1.3. (3) AVOID RANGE OF WEAPON

1.1.2.1.3.1. (10) BY DETECTING FIRING SOURCE OUTSIDE ITS RANGE

10	6	0
WELL OUTSIDE	JUST OUTSIDE	WITHIN RANGE

1.1.2.1.3.2. (3) BY FREEDOM TO MANEUVER

10	5	0
COMPLETE FREEDOM OF MANEUVER	LIMITED MANEUVER ROOM	NOT FREE TO MANEUVER

1.1.2.1.3.3. (5) BY SUPERIOR SPEED AND MANEUVERABILITY

10	2	0
SUPERIOR	EQUAL	ENEMY SUPERIOR

1.1.2.2. (5) AVOID DAMAGE FROM AIR-TO-SURFACE WEAPONS

1.1.2.2.1. (10) SHIELD FROM EFFECTS OF WEAPONS

1.1.2.2.1.1. (10) BY USE OF TERRAIN FOR COVER

10	7	5	3	1	0
COMPLETELY COVERED	>75% COVERED	50 > 75% COVERED	25 > 50% COVERED	10 > 25% COVERED	COMPLETELY EXPOSED

1.1.2.2.1.2. (2) BY USE OF PROTECTIVE ARMOR

10	6	2	0
ARMOR FRONT TO ENEMY LINE OF FIRE	ARMOR FRONT TO > 50% ARC OF FIRE	GREATEST VULNERABILITY TO >50% ARC OF FIRE	GREATEST VULNERABILITY EXPOSED

1.1.2.2.1.3. (3) BY USE OF PERSONNEL PROTECTIVE MEASURES

1.1.2.2.1.3.1. (10) BY "BUTTONING-UP" TANK

10	0
"BUTTONED-UP"	HATCHES OPEN

1.1.2.2.1.3.2. (3) BY PROTECTIVE CLOTHING/GAS MASK

10	0
YES	NO

1.1.2.2.1.4. (5) BY USE OF TERRAIN FOR PRE-MATURE FUSE DETONATION

10	5	0
DENSE OVER-FOILAGE	MODERATE OVERFOILAGE	NO OVER-FOILAGE

1.1.2.2.2. (5) IMPAIR ACCURACY OF WEAPON

1.1.2.2.2.1. (10) BY COUNTER-FIRE

10	0
GUIDANCE SOURCE WITHIN OWN WEAPON RANGE	GUIDANCE SOURCE BEYOND OWN WEAPON RANGE

1.1.2.2.2.2. (3) BY ELECTRONICS/IR COUNTERMEASURES/DECOY

10	0
WEAPON SUSCEPTIBLE TO COUNTERMEASURES	WEAPON NOT SUSCEPTIBLE

1.1.2.2.2.3. (5) BY "JINKING" MANEUVERS

10	0
EFFECTIVE	NOT EFFECTIVE

1.1.2.2.3. (3) AVOID RANGE OF WEAPON

1.1.2.2.3.1. (10) BY DETECTING FIRING SOURCE OTUSIDE ITS RANGE

10	1	0
WELL OUTSIDE	JUST OUTSIDE	WITHIN RANGE

1.1.2.2.3.2. (5) BY FREEDOM TO MANEUVER

10	5	0
COMPLETE FREEDOM OF MANEUVER	LIMITED MANEUVER ROOM	NOT FREE TO MANEUVER

1.1.3 (5) AVOID DAMAGE FROM OWN ACTION

1.1.3.1. (10) AVOID DAMAGE FROM TERRAIN

1.1.3.1.1. (10) AVOID STEEP DECLINE/INCLINE

10	7	3	0
LEVEL TERRAIN	GENTLY ROLLING TERRAIN	HILLY	STEEP CLIFFS

1.1.3.1.2. (3) AVOID COLLISION WITH OBSTRUCTIONS

10	7	3	0
CLEAR OPEN TERRAIN	OCCASIONAL ROCK/TREE OBSTRUCTION	PROFUSELY ROCKY, FORRESTED	DIRECTION BLOCKED BY OBSTRUCTION

1.1.3.1.3. (5) AVOID DEEP WATER/MUD/SNOW

10	7	3	0
SOLID, SMOOTH SURFACE CONDITIONS	TRACTION OCCASIONALLY SLOWED BY BY MUD/SNOW	PASSABLE BUT SPEED/ MANEUVER SIGNIFICANTLY IMPEDED	IMPASSABLE

1.1.3.2. AVOID DAMAGE FROM FRIENDLY FORCES

1.1.3.2.1. (3) BY COLLISION

10	7	3	0
NO FRIENDLY FORCE IN NEAR VICINITY	TRAFFIC WELL-DISPERSED AND CONTROLLED	HEAVY TRAFFIC	"TRAFFIC JAM" CONGESTION

1.1.3.2.2. (10) BY MOVEMENT INTO FIELD OF FIRE

10	3	3
MOVEMENT WELL CLEAR OF FIELD OF FIRE	SOME RISK OF DAMAGE FROM INACCURATE FALL OF SHOT	IN DIRECT PATH OF FIELD OF FIRE

1.1.3.2.3. (5) BY MOVEMENT INTO FRIENDLY EMPLACED OBSTACLES

10	0
AVOID	FAIL TO AVOID

1.1.3.3. (3) AVOID DAMAGE FROM ENEMY EMPLACEMENTS

1.1.3.3.1. (10) AVOID MINEFIELDS

10	0
AVOID	FAIL TO AVOID

1.1.3.3.2. (5) AVOID TANK TRAPS/OBSTACLES

10	0
AVOID	FAIL TO AVOID

1.1.4. (3) PROVIDE PHYSICAL SUSTENANCE

1.1.4.1 (5) SUSTAIN PERSONNEL

1.1.4.1.1. (10) MAINTAIN PHYSICAL STRENGTH

1.1.4 1.1.1. (10) PROVIDE NOURISHMENT (FOOD AND WATER)

10	7	5	2	0
RECENTLY AND ABUNDANTLY PROVIDED	PROVIDED WITHIN LAST SIX HOURS	PROVIDED WITHIN LAST 12 HOURS	THIRST AND HUNGER DISTRACTING EFFICIENCY	WEAKENED BY PROLONGED THIRST/HUNGER

1.1.4.1.1.2. (3) PROVIDE ADEQUATE REST

10	7	5	2	0
WELL RESTED	ADEQUATE STAMINA FOR NEXT 4-6 HOURS	ADEQUATE STAMINA FOR NEXT 2-4 HOURS	FATIGUE DETRACTING FROM STAMINA	WEAK FROM FATIGUE

1.1.4.1.1.3. (5) AVOID EXTREMES OF HEAT OR COLD

10	5	2	0
MODERATE AMBIENT TEMPERATURES	HEAT OR COLD A MINOR DISCOMFORT	HEAT OR COLD SIGNIFICANTLY DISCOMFORTING	AT THRESHOLD OF INCAPACITATION FROM HEAT OR FREEZING

1.1.4.1.2. (5) MAINTAIN MENTAL ALERTNESS

10	5	2	0
KEENLY ALERT AND ATTENTIVE	GENERALLY ALERT OCCASIONAL LAPSE OF ATTENTION	DROWSY AND INATTENTIVE	ASLEEP

1.1.4.2. (10) SUSTAIN EQUIPMENT

1.1.4.2.1. (5) MAINTAIN ADEQUATE AMMUNITION SUPPLY

10	8	6	4	2	0
LOADED TO CAPACITY	>75%	75 > 50%	50 > 25%	<25%	OUT OF AMMUNITION

1.1.4.2.3. (10) MAINTAIN ADEQUATE FUEL SUPPLY

10	8	6	4	2	0
FUELED TO CAPACITY	>75%	75 > 50%	50 > 25%	<25%	OUT OF FUEL

1.1.4.2.3. (3) MAINTAIN EQUIPMENT OPERABILITY

10	7	5	2	0
FULL SYSTEMS CAPABILITY	OPERATIONALLY READY WITH MINOR SYSTEM DEGRADATION	OPERATIONALLY READY WITH SIGNIFICANT DEGRADATION	NOT OPERATIONALLY READY BUT MOBILE	NOT READY NOT MOBILE

1.2 (5) ENEMY DESTRUCTION

1.2.1. (10) DESTROY ENEMY BY OWN FIREPOWER

1.2.1.1. (5) FIRE WEAPON WITHIN EFFECTIVE RANGE

10	5	2	0
WITHIN MOST EFFECTIVE RANGE	AT MAXIMUM EFFECTIVE RANGE	AT MAXIMUM RANGE	BEYOND RANGE

1.2.1.2. (2) FIRE WEAPON FOR MAXIMUM LETHALITY (OPTIMAL AIM POINT)

10	5	0
LINE OF FIRE TO ENEMY'S GREATEST VULNERABILITY	LINE OF FIRE TO ENEMY'S MEDIAN VULNERABILITY	LINE OF FIRE TO ENEMY'S GREATEST STRENGTH

1.2.1.3. (3) FIRE WEAPON FOR MAXIMUM ACCURACY

10	5	0
CAREFUL AIM OPTIMUM FIRE CONTROL SOLUTION	LESS THAN OPTIMUM FIRE CONTROL SOLUTION	NO FIRE CONTROL SOLUTION

1.2.1.4. (10) SHOOT FIRST

10	5	0
FIRST	SIMULTANEOUSLY	ENEMY SHOOTS FIRST

1.2.2. (5) DESTROY ENEMY BY SUPPORT FORCE FIREPOWER

1.2.2.1. (10) BY ARTILLERY SUPPORT

10	5	3	1	0
DEDICATED, DIRECT SUPPORT IMMEDIATELY ON CALL	DIRECT SUPPORT 5-MINUTE RESPONSE	GENERAL SUPPORT 15-MINUTE RESPONSE	GENERAL SUPPORT 30-MINUTE RESPONSE	NOT AVAILABLE

1.2.2.2. (3) BY INFANTRY SUPPORT

10	3	1	0
AVAILABLE IN FAVORABLE POSITION FOR IMMEDIATE RESPONSE	AVAILABLE WITH 15-MINUTE DELAY	AVAILABLE WITH 30-MINUTE DELAY	NOT AVAILABLE

1.2.2.3. (5) BY ATTACK HELICOPTER SUPPORT

10	5	3	1	0
AVAILABLE IMMEDIATELY ON CALL	AVAILABLE WITH 5-MINUTE RESPONSE	AVAILABLE WITH 10-MINUTE RESPONSE	30-MINUTE DELAY	NOT AVAILABLE

1.2.2.4. (2) BY CLOSE AIR SUPPORT

10	5	3	1	0
AVAILABLE IMMEDIATELY ON CALL	AVAILABLE WITH 5-MINUTE RESPONSE	AVAILABLE WITH 10-MINUTE RESPONSE	30-MINUTE DELAY	NOT AVAILABLE

1.2.3. (2) DESTROY OR NEUTRALIZE ENEMY BY ENTRAPMENT

1.2.3.1. (10) LURE INTO MINEFIELD

10	5	2	0
MINEFIELD IN DIRECT PATH OF ENEMY ADVANCE	MINEFIELD ASTRIDE LOGICAL ROUTE WITH SOME DIVERGENCE FROM PATH OF ADVANCE	MINEFIELD WOULD REQUIRE SIGNIFICANT DIVERGENCE FROM PATH OF ADVANCE	NO MINEFIELD EMPLACED

1.2.3.2. (3) LURE INTO OPEN, EXPOSED POSITION

10	7	5	2	0
IN DIRECT PATH OF ENEMY ADVANCE	IN ROUTE TO PROBABLE OBJECTIVE	IN ROUTE TO DESIRABLE OBJECTIVE OF OPPORTUNITY	DIVERSION FROM OBJECTIVE REQUIRED	NOT FEASIBLE WITH SURROUNDING TERRAIN

1.2.3.3. (5) LURE INTO PREPARED OBSTACLE

10	7	2	0
OBSTACLE IN OPTIMAL INTERPOSED POSITION	OBSTACLE IN GENERAL PATH OF ENEMY ADVANCE	OBSTACLE OFF GENERAL PATH OF ADVANCE	NO OBSTACLE EMPLACED

1.2.3.4. (2) LURE INTO NATURAL OBSTRUCTION OR DANGER

10	7	2	0
OBSTACLE IN OPTIMAL INTERPOSED POSITION	OBSTACLE IN GENERAL PATH OF ENEMY ADVANCE	OBSTACLE OFF GENERAL PATH OF ADVANCE	NO NATURAL OBSTRUCTION

1.2.4. (3) DESTROY OR NEUTRALIZE ENEMY BY DENYING SUSTENANCE

1.2.4.1. (10) DESTROY OR BLOCK SOURCE OF AMMUNITION SUPPLY

10	7	3	0
SOURCE OF SUPPLY IN SIGHT AND IN RANGE OF WEAPON	SOURCE OF SUPPLY ACCESSIBLE WITH MOVEMENT OF LOW VULNERABILITY	SOURCE OF SUPPLY ACCESSIBLE WITH MOVEMENT OF MODERATE VULNERABILITY	SOURCE OF SUPPLY UNKNOWN OR INVULNERABLE TO OWN WEAPON

1.2.4.2. (7) DESTROY OR BLOCK SOURCE OF FUEL SUPPLY

10	7	3	0
SOURCE OF SUPPLY IN SIGHT AND IN RANGE OF WEAPON	SOURCE OF SUPPLY ACCESSIBLE WITH MOVEMENT OF LOW VULNERABILITY	SOURCE OF SUPPLY ACCESSIBLE WITH MOVEMENT OF MODERATE VULNERABILITY	SOURCE OF SUPPLY UNKNOWN OR INVULNERABLE TO OWN WEAPON

1.2.4.3. (2) DESTROY OR BLOCK ACCESS TO MAINTENANCE SUPPORT

10	7	3	0
SOURCE OF SUPPORT IN SIGHT AND IN RANGE OF WEAPON	ACCESS ROUTE VULNERABLE TO INTERDICTION FROM COVERED POSITION	ACCESS ROUTE VULNERABLE TO INTERDICTION FROM POSITION OF MODERATE VULNERABILITY	NEITHER SOURCE NOR ACCESS ROUTE ACCESSIBLE

1.2.4.4. (3) ENTICE INEFFECTIVE AMMUNITION EXPENDITURE

10	7	5	3	0
RAPID FIRE AT PHANTOM TARGETS WITH HARMLESS FALL OF SHOT	FIRE BEYOND EFFECTIVE RANGE OF WEAPONS	FIRE WITH LOW PROBABILITY OF HIT	FIRE WITH LOW PROBABILITY OF DAMAGE	FIRE WITH HIGH PROBABILITY OF KILL

1.2.4.5. (3) ENTICE INEFFECTIVE FUEL EXPENDITURE

10	5	0
HIGH CONSUMPTION RATE WITH NO FORWARD PROGRESS TOWARD OBJECTIVE	HIGH CONSUMPTION RATE WITH LITTLE FORWARD PROGRESS TOWARD OBJECTIVE	LOW CONSUMPTION RATE WITH NO WASTED MOVEMENT VIS-A-VIS OBJECTIVE

1.2.4.6. (2) DISSIPATE PERSONNEL STRENGTH

5	0
HIGH ENERGY EXPENDITURE NO OPPORTUNITY FOR REST. HIGH SENSORY DISCOMFORT	LOW ENERGY DEMAND WELL RESTED NO SENSORY DISCOMFORT

TABLE 2

ALLOCABLE RESOURCES

TANK

- MANEUVERABILITY
- ARMOR SHIELD
- SMOKE GRENADE LAUNCHER
- SMOKE GENERATOR
- FUEL STATE (ENDURANCE)
- MAINTENANCE STATE

WEAPONS

- MAIN ARMAMENT
- COAXIAL WEAPON
- LOADER'S WEAPON
- COMMANDER'S WEAPON
- AMMUNITION STATE (ENDURANCE)
- AMMUNITION STATE (ENDURANCE)

PERSONNEL

- PHYSICAL ENDURANCE
- MENTAL ENDURANCE
- LEVEL OF NBC PROTECTIVE READINESS

EXTERNAL COMMUNICATIONS

- REPORT SITUATION
- REQUEST SUPPORTING ARMS
- REQUEST LOGISTICS SUPPORT

FIGURE 1

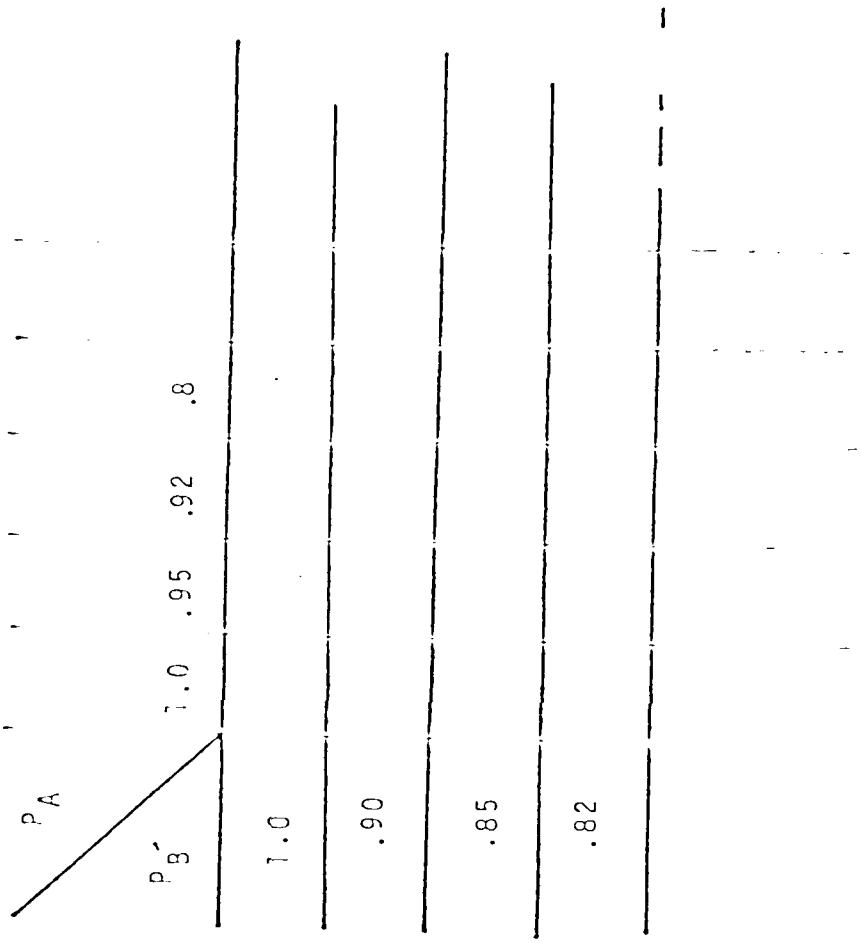
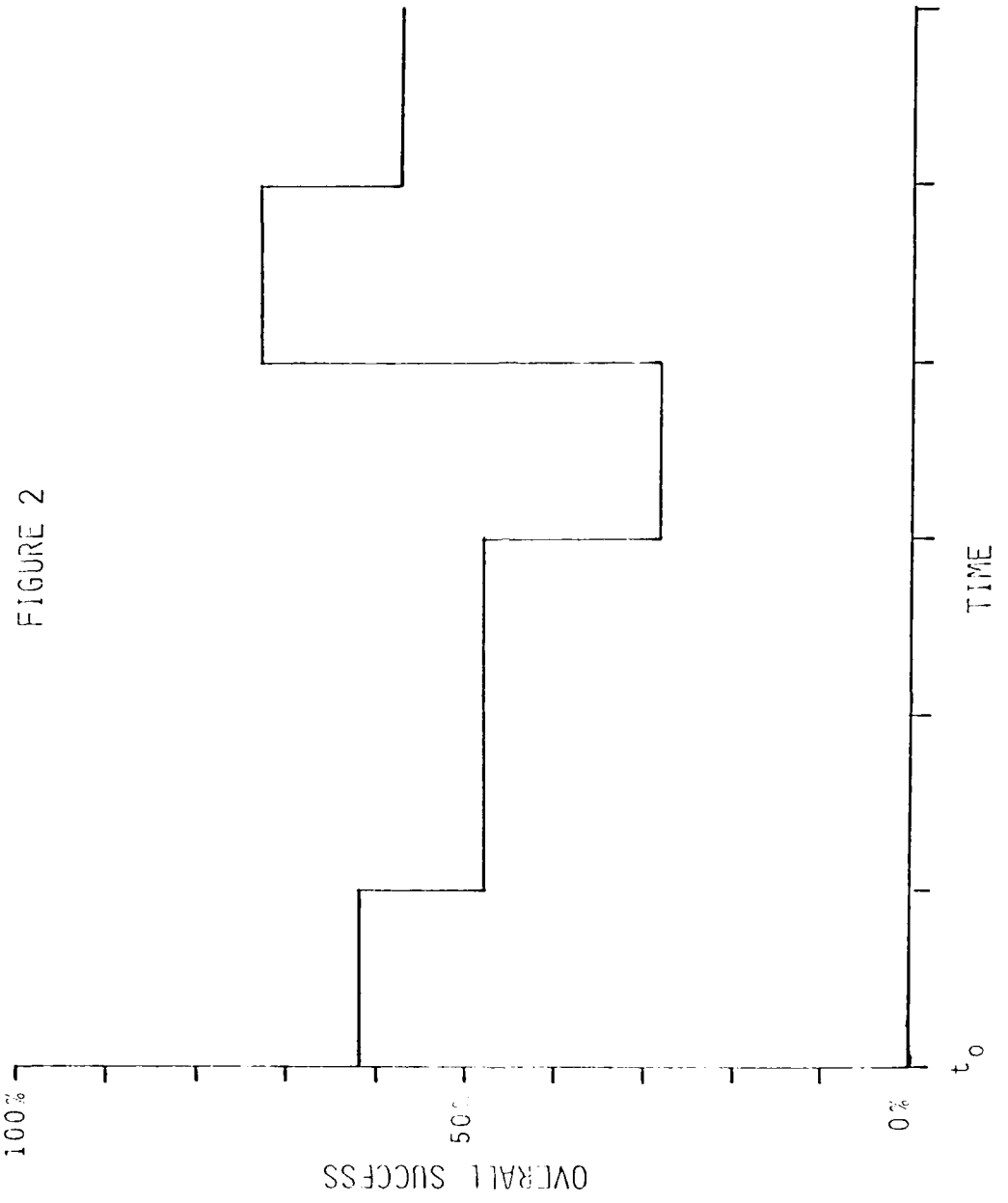


FIGURE 1

FIGURE 2



APPENDIX B

SECOND PROGRESS REPORT

CONTRACT NO. MDA903-81-C-0509

DECISION SCIENCE, INC.
4901 MORENA BOULEVARD
SAN DIEGO, CALIFORNIA
92117 (714) 273 2922

82-413-2

THE ADAPTIVE MANEUVERING LOGIC IN TANK WARFARE SIMULATION

SECOND PROGRESS REPORT

Contract No. MDA903-81-C-0509

Major Jack Thorpe, USAF
Defense Advanced Research Projects Agency
Cybernetics Technology Division
1400 Wilson Boulevard
Arlington, VA 22209

March 25, 1982

TABLE OF CONTENTS

INTRODUCTION

This second quarterly R & D Status Report covers the work performed by Decision Science, Inc. under Contract Number MDA903-81-C-0509 during the period 12 January 1982 through 10 March 1982. The scope of work of the total nine-month term of this contract effort is to include consultation with experts in main battle tank operations; the construction of a valuated state space for defining the purpose and measuring achievement of the tank commander in combat; and the creation of the architecture, flow diagrams and initial software specifications for a computer program for controlling simulated tank maneuvers according to the dictates of the valuated state space as applied by the Adaptive Maneuvering Logic (AML). The objective of the work is to demonstrate the feasibility of a computer program for tank warfare simulation that incorporates the AML technique to provide an intelligently interactive opponent tank in simulated ground combat engagement with a human (trainee) controlled tank. The AML adaptation for ground combat simulation would lend a significant new dimension in the field of computer-generated simulation.

The First Progress Report, hand delivered and briefed at DARPA Headquarters on 12 January 1982, described the underlying concept of the hierarchic valuated state space and provided an illustrative example of a valuated state space of the Tank Commander's Purpose in combat. The latter

was constructed on the basis of interaction with tank warfare experts at the U.S. Army Armor School, Fort Knox, and documentation provided through those sources as well as sources at Army Headquarters, the Pentagon, and at the Defense Intelligence Agency. As further described in the First Progress Report, this valuated state space, in conjunction with a grid system to delineate small, discrete sectors of the combat engagement area, provided the basis for evaluating candidate positions for the AML Tank vis-a-vis the opponent or Trainee Tank. An optimum candidate grid position was then selected. Work during the recent period concentrated on the development and testing of an algorithm for determining the "best" path for reaching the selected "best" candidate position. The Discussion which follows describes the evolution of the algorithm and provides an illustrative example of its application.

DISCUSSION

Having selected the "best" candidate grid position for the AML Tank to occupy relative to the present grid position of the Trainee Tank, it then becomes necessary to select the "best" path for the AML Tank to traverse to gain that position. Just as the best candidate position was selected on the basis of optimizing the AML Tank's utility, so, too, can a "best" path be selected on the basis of Utility Function optimization. There is a difference, however, in that the Utility Function of the path must appropriately aggregate the individual utility values of all of the grid squares throughout the course of the path. It must be sensitive to opportunities to improve the defensive or offensive posture of the AML Tank en route to the destination, avoid adverse features of terrain, and allow for circuitous routing that may well extend outside the area encompassing both the initial and the destination grid positions. Additionally, the overall Utility of a path must be particularly sensitive to any high risk grid squares through which it passes and impart the appropriate degree of aggressiveness to the AML Tank according to the tank commander's mission or purpose as defined by the Valuated State Space (i.e., relative importances accorded those parameters of "own survival" vis-a-vis those of "enemy destruction").

Throughout the period of this report, a major effort has been directed toward the development and testing of an algorithm for path selection that incorporates each of the foregoing considerations. The algorithm draws on the Valuated State Space of the Tank Commander's Purpose to derive utility measures of "survivability" and "attackability" for the AML Tank at each grid square along successively expanding paths terminating at the destination grid square. It then combines these measures to determine a single utility value for each alternative path at each expansion and selects as the "best" path that path of highest utility value. The process is repeated as the area of investigation is incrementally expanded, square by square, outward from the destination grid square until encompassing the AML Tank's present position or point of origin. Finally, the area is expanded sufficiently beyond the point of path origin to allow for the possibility that the best path may lead initially in a direction opposite or away from the point of destination. The algorithm can perhaps be best described by its step-by-step application in an illustrative example.

Figure 1 depicts a combat engagement area with grid reference system superimposed. Note that there are twenty grid squares horizontally across the x-axis and sixteen grid squares vertically up the y-axis. Grid squares are identified by numbers separated by comma, referring first to column from left to right (west to east) across the x-axis and then to row from bottom to top (south to north) up the y-axis.

The grid square at the lower-left corner, for example, is identified as (1,1); the grid square at the upper-right corner (20,16), the grid square at the center (10,8), and so forth. The tank symbol at (9, 13) represents our "opponent" tank (Trainee Tank) who has pinned down a squad of "friendly" troops in a moderately forrested area in and around grid squares (12,9) and (13,10). The initial position of the "friendly" tank (AML Tank) is shown at Point A just to the left of grid square (1,5). The tank symbol at square (17, 13) indicates the candidate grid position selected by AML Program as the optimum position for the AML Tank relative to the Trainee Tank's position. The problem now is to determine the most advantageous or "best" path for the AML Tank to follow in traversing from Point A to Point B.

Before demonstrating how the algorithm is applied to determine the best path, some further explanation of the battlefield terrain depicted in Figure 1 may be in order. Immediately to the left of the Trainee Tank is a steep cliff (impassable). Atop this cliff in and immediately surrounding grid squares (5, 13), (6, 12) and (6, 13), is a plateau area sloping off less severely (passable) to the left (west). Ringing the area to the right from grid squares (1,6) and (1,7) curving first easterly then northerly and finally westerly to grid square (12, 16) is a river, impassable for crossing but affording a covered, trafficable route immediately between the bank and riverbed. Trees line and overhang both banks of the river providing cover and concealment from the

air. Contour lines generally surrounding grid squares (10, 15); (11, 6); and (16, 13) represent hills that would provide cover from ground fire originating from the opposite sides. The symbols in the vicinity of grid squares (5, 8), (6, 6) and (7, 6) depict brush and foliage offering some partial concealment from both ground and air detection. The curved, broken line describing an arc centered at the position of the Trainee Tank indicates the effective lethal range of its main armament.

As noted earlier, the algorithm for best path determination combines the utility measures of "survivability" and "attackability" for all grid squares along a path to determine a single utility value for the path as a whole. Referring to Figure 2, the utility measures for "survivability," "attackability" and overall Utility are shown for each grid square (reading from top to bottom within the square, respectively) as evaluated from the Valuated State Space for that particular grid square, assuming it were the position of the AML Tank relative to the Opponent or Trainee Tank at its present position (i.e., grid square (9, 13). The overall Utility for each grid square is based on importance weights for "survivability" and "attackability" of 10 and 5, respectively. Recall that these utility measures are derived from the Valuated State Space of the Tank Commander's Purpose, as described in the previous Progress Report dated 12 January 1982. (The utility measure of "survivability" derives from those parameters of purpose included under "Own Survival"--"attackability" from those parameters included under "Enemy

Destruction"). Although a key determining factor in the selection of the best candidate grid position or destination, the overall utility value for each grid square does not, per se, figure into the calculus of overall path utility. But rather, the latter is calcualted according to the formula:

$$U = K_1\left(1 - \prod_{i=1}^{n} DF_{T_i}\right) + K_2 \prod_{i=1}^{n} DF_{A_i}$$

where U is the Utility of the path;

$\prod_{i=1}^{n} DF_{A_i}$ is the product of the utility measures of Defensive Posture of the AML Tank ("survivability" value) at each of the n grid points along the path;

$\prod_{i=1}^{n} DF_{T_i}$ is the product of the utility measures of Defensive Posture of the Trainee Tank (1-AML Tank's "attackability" value) at its stationary position relative to each of the n grid points along the AML Tank's path;

K_1 is the relative importance weight assigned to enemy destruction or "attackability"; and

K_2 is the relative importance weight assigned to own survival or "survivability" according to the Valuated State Space of Tank Commander's Purpose.

With the utility of any given path determined according to the above formula, the next step in the algorithm is the derivation of that path of highest utility from point of origin to point of destination. This is accomplished by

calculating the utility of paths to Point B from all points on successively expanded squares about the point of destination, Point B, until encompassing the point of origin, Point A; and then continuing the expansion procedure sufficiently to locate or foreclose the possibility of the path of highest utility being one that moves initially in a direction away from Point B.

To illustrate, Figures 3 through 30 show the first three expansion sequences. Referring to Figure 3, the first or smallest square of grid squares about Point B is outlined by the heavy black line. This square encompasses grid squares (18, 13), (18, 12), (17, 12) and (17, 13) of the combat engagement area depicted in Figure 1. Numbers appearing between vertices are the utility measures of Defensive Posture of the Trainee and AML Tank, respectively, (DF_T and DF_A) corresponding to the "attackability" and "survivability" measures for that grid square. To facilitate identification of vertices, a new coordinate system will be used with Point B at the origin, x positive to the right and y positive up. Vertex (1, 1) is then the upper right-hand corner of the outlined square; (0,1) the upper center vertex; (-1, -1) the lower left-hand corner, and so forth. At this first square, there are eight points (vertices) from which the utility of the path from that point to Point B is calculated according to the formula:

$$U = K_1\left(1-\prod_{i=1}^{n} DF_{T_i}\right) + K_2 \prod_{i=1}^{n} DF_{A_i}$$

From each point or vertex there is one path of highest utility. Each such path is determined as follows:

(1) Select one of the center vertices leading directly in toward Point B, for example (1,0).

(2) Calculate the Utility of the direct path to Point B. Record the Utility value, the products, $\prod^{n} DF_T$ and $\prod^{n} DF_A$,* and indicate the direction of the path by arrowhead, ◀, as shown in Figure 4.

(3) Proceeding counterclockwise around Point B, repeat the procedure for each of the remaining three vertices leading directly in toward Point B, as shown in Figure 5.

(4) Starting with the first vertex in a counterclockwise direction from the originally selected vertex, in this case corner vertex (1, 1), calculate the Utility of the path from that vertex leading in a clockwise direction. Record the Utility value, the products, $\prod DF_T$ and $\prod DF_A$, and direction of the path by arrowhead, ▼ . See Figure 6.

(5) Proceeding to the next adjacent counterclockwise vertex, in this case vertex (0, 1), calculate the Utility of the path from that vertex leading in a clockwise direction. Compare the Utility value of the clockwise path with that of the recorded (in other words, inward) path, select the path of highest utility, and retain or change the recorded Utility value, products, $\prod DF_T$ and $\prod DF_A$, and

*Note that in this instance n is equal to one.

arrowhead direction as appropriate. In the case of the illustrative example, the Utility of the recorded inward path exceeds that of the clockwise path. As shown in Figure 7, the clockwise path is therefore eliminated, and the Utility value, products ΠDF_T and ΠDF_A, and arrowhead direction ↓ of the inward path are retained as the recorded path from that vertex. Continue the procedure moving successively to the next counterclockwise vertex, checking the path in a clockwise direction until the Utility value of the clockwise path from each vertex has been calculated and recorded at the corner vertices, together with products, ΠDF_T and ΠDF_A, and arrowhead direction, or compared with the presently recorded Utility values at the center vertices and the recorded Utility values, products and arrowhead direction at those vertices retained or changed as appropriate to indicate the higher Utility of the two paths compared. See Figure 8.

(6) Starting with the first vertex in a clockwise direction from the originally selected vertex, in this case corner vertex (1, -1), calculate the Utility of the path from that vertex leading in a counterclockwise direction. Compare the Utility value of the counterclockwise path with that of the presently recorded path (in this case the clockwise path),

select the path of highest Utility, and retain or change the recorded Utility value, products, ΠDF_T and ΠDF_A, and arrowhead direction as appropriate. See Figure 9. Continue the procedure, moving successively to the next clockwise vertex, checking the path in a counterclockwise direction until the counterclockwise path from each vertex has been checked, and the recorded Utility value, products and arrowhead at each vertex retained or changed as appropriate to indicate the highest Utility path from that vertex to Point B. See Figure 10.

Note that in the case of the illustrative example (still referring to Figure 10), the Utility of the counterclockwise path from the first vertex checked, vertex (1, -1) exceeds that of the "presently recorded" clockwise path from that vertex. The clockwise path is therefore eliminated in favor of the counterclockwise path. The Utility value, products ΠDF_T and ΠDF_A, and arrowhead direction are changed to those of the counterclockwise path which then become the "presently recorded" path values and direction for any subsequent comparisons at that vertex (until or unless some subsequent comparison should dictate another change) and for calculating the Utility values of any path leading to it. No change occurs at vertex (0, -1) where the inward path is retained, nor at vertex (-1, -1) where the clockwise path is retained. Moving to the next clockwise vertex, vertex (-1, 0),

it is seen that the counterclockwise path from that vertex would lead directly against the arrowhead at the next vertex, it having been determined that the best path from that vertex is the clockwise path leading to vertex (-1, 0). Obviously, then, the presently recorded path from vertex (-1, 0) is better than one leading initially in a counterclockwise direction since the latter would only result in being turned back to its point of origin upon arrival at the next vertex. No calculation of path Utility is necessary in such instances to make that determination. It is, then, a general rule of the algorithm that a path from any vertex that would lead in a direction against the recorded arrowhead of a next adjacent vertex can be eliminated a priori.

Returning to the illustrative example and continuing the sequence of moving to the next clockwise vertex, checking the counterclockwise path to the vertex just checked, changes of the "presently recorded" values and directions to those of the counterclockwise path occur at vertices (-1, 1) and (1, 1).

Note that in this illustrative example there were no direction changes resulting from a clockwise or counterclockwise path having a higher Utility value than the direct, inward path. This may not always be the case, and, in fact, will not be the case in subsequent expanded squares as will be shown later in this illustrative example. Whenever such a change occurs, then the Utility values and products, ΠDF_T and ΠDF_T of all paths affected by that change must be recomputed and compared with presently recorded alternative

paths, the Utility values of which may no longer be superior to the recomputed values. Where inversions occur, new path directions and Utility values will ensue which, in turn, must be checked for their possible effect on still other paths. For example, referring to Figure 11, let us assume that the Utility value of the counterclockwise path from vertex (1, 0) were higher than the utility of the inward path as shown, resulting in a direction change at that vertex from inward to counterclockwise. The Utility value of the path from vertex (1, -1) is now based on the products, ΣDF_T and ΣDF_A of the counterclockwise path rather than the inward path from vertex (1, 0) and is increased as shown. The counterclockwise path from vertex (0, -1) must now be recomputed on the basis of the change in values of the path from vertex (1, -1). The recomputed Utility value of the counterclockwise path being higher than the Utility value of the inward path results in a direction change at vertex (0, -1) from inward to counterclockwise. This in turn will change the Utility value of the counterclockwise path from vertex (-1, -1) which must therefore be recomputed and checked against the presently recorded Utility value of the clockwise path from that vertex. Since the clockwise path has the higher Utility value, no change occurs at vertex (-1, -1), and, since there are at this time no outer square transitions to be considered, no further checks as a result of the change at vertex (1, 0) are required. The foregoing sequence of checks can be stated as a second general rule of the algorithm, the full extent of which will

become more apparent as the area of path exploration is extended outward by subsequent square expansions. That rule may be stated as follows: Any vertex at which a change in either Utility value or direction has occurred must have the immediately surrounding vertices (exclusive of the vertex to which the changed path now leads since that direction has been checked) checked to see whether or not those surrounding vertices now require a direction and/or Utility value change.

Returning to our illustrative example, the best path from each of the vertices of the grid squares immediately surrounding the destination of the AML Tank, Point B, has now been determined. These are shown with their associated Utility values and DF_T/DF_A products in Figure 12. A caveat must be added here, however, since possible path excursions outside the four immediately surrounding grid squares have not yet been investigated. The term "best path" should therefore more accurately be qualified to read "best path which does not extend beyond the outer perimeter of the four grid squares immediately surrounding the destination point." How the algorithm in its complete form allows for outward path excursions will be shown as the squares of investigation are expanded outward from Point B.

The algorithm will next be applied to determine the best path from each vertex on the first expanded square around Point B, encompassing sixteen grid squares of the combat engagement area:

(1) Calculate the Utility of each path leading directly inward from the vertices of the expanded square to the corresponding vertices of the inner square. Record the Utility value, the products, ΠDF_T and ΠDF_A, and the path direction by arrowhead, ↓, as shown in Figure 13. The double lines appearing below and to the left of vertices (2, 1) and (2, 2) indicate impassable terrain barring paths to or from those vertices and adjacent vertices within the expanded square.

(2) Starting at the vertex of highest Utility, that is, vertex (-2, 0) in this case, move one vertex in a counterclockwise direction and calculate the Utility value of the path leading from that vertex in a clockwise direction. Compare the Utility of the clockwise path with that of the recorded (that is, inward,) path, select the path of highest Utility, and retain or change the recorded values and arrowhead direction as appropriate. If a change occurs, note and record for further reference the adjacent vertices that must be checked for recomputed path Utility values and possible direction changes pursuant to the second general rule of the algorithm. See Figure 14.

(3) Proceeding to the next adjacent vertex in a counterclockwise direction, calculate the Utility value of

the path leading from that vertex in a clockwise direction. Compare and select the path of highest utility as in Step (2) above. Continue the procedure, moving successively to the next counterclockwise vertex, checking the path in a clockwise direction until the clockwise path from each vertex has been checked. See Figure 15.

(4) Starting again at the same vertex as in Step (2), move one vertex in clockwise direction and check the Utility of the path leading from that vertex in a counterclockwise direction. If the counterclockwise path is opposed by the arrowhead indicating direction of the presently recorded path from the next adjacent counterclockwise vertex, then the first general rule of the algorithm applies, and the counterclockwise path can be eliminated without further computation. If not opposed by the arrowhead at the next adjacent counterclockwise vertex, calculate the Utility value of the counterclockwise path, compare that Utility value with Utility value of the presently recorded path, select the path of highest Utility, and retain or change the recorded values and arrowhead direction as appropriate. Again, in accordance with the second general rule of the algorithm, wherever changes occur, note and record the adjacent vertices that must be checked for Utility and possible path redirection as a result of the change.

(5) Proceeding to the next adjacent vertex in a clockwise direction, repeat the procedure of Step (4) above, moving successively to each next clockwise vertex, checking the path in a counterclockwise direction until the counterclockwise path from each vertex has been checked. See Figure 16.

At this point of the investigation, the algorithm has determined the best path to Point B from each of the vertices of the first expanded square exclusive of any possible outward excursions (which will be explored at the next larger expanded square) or any possible redirection from the inner square to the first expanded square en route to Point B which will be explored by the next step of the algorithm. Figure 17 shows the currently determined "best" paths subject to the exclusions just stated.

Recall that in conducting the clockwise/counterclockwise checks of Steps (2) through (5), whenever a change in direction and Utility value occurred at any vertex, the adjacent vertices requiring re-examination in accordance with the second general rule of the algorithm were recorded for future reference. Referring to Figure 16, note that changes occurred at vertices (-2, -1), (1, 2), (2, -1), (2, -2), (1, -2) and (0, -2). The next step of the algorithm, then, may be stated as follows:

(6) Referring to the record of vertices adjacent to those where changes occurred in Steps (2) through (5), calculate the Utility value of the path

leading from each recorded vertex to its adjacent, changed vertex. Compare that Utility value with Utility value of the presently recorded path, select the path of highest utility, and retain or change the recorded values and arrowhead direction as appropriate. If a change occurs, note and record the additional adjacent vertices that must be checked for Utility and possible path redirection as a result of the change.

To illustrate the application of Step (6), the vertices where changes occurred in Steps (2) through (5) are indicated in Figure 18 by a small circle at the base of the arrowhead. The change at vertex (-2, -1) requires that the Utility of the path from adjacent vertex (-1, -1) be checked to determine whether the path through vertex (-2, -1) may not be better than the presently recorded path. Since the Utility value of the path through vertex (-2, -1) calculates to be less than that of the presently recorded path, no change occurs at vertex (-1, -1) and no further checks are required as a result of the change at vertex (-2, 1). See Figure 19. Similarly, checks at vertices (1, 1) and (0, -1) required by direction changes at vertices (1, 2) and (0, -2), respectively, fail to disclose a better path than the presently recorded one and no further checks are required as a result of either of those changes. Checking the vertex (1, -1) required by the direction change at vertex (2, -1), however, results in a path Utility and direction change, which, in turn, requires checks at vertices (0, -1) (1,-2) and (1,0). Accordingly, these vertices

are recorded for subsequent examination. Continuing on to the change at vertex (2, -2) no additional checks are required since the Utility and direction at vertex (1, -2) were determined sequentially after those of vertex (2, -2) during the counterclockwise path checks of Step (5). The changes at vertices (1, -2) and (0, -2) require checks at vertices (1, -1) and (0, -1), respectively, neither of which discloses a better path than those presently recorded; hence, no further checks are required as a result of these two changes.

(7) Referring to the vertices recorded for subsequent examination because of Utility and/or direction changes occurring during Step (6), repeat the procedure of Step (6) to determine whether any additional changes are required.

Applying Step (7) to our illustrative example, recall that in Step (6), vertices (0, -1), (1, -2) and (1, 0) were recorded for subsequent examination as a result of the change occurring at vertex (1, -1). Accordingly, the Utility of the path from each of these vertices through vertex (1, -1) must be calculated and compared with the Utility of the presently recorded path. As shown in Figure 20, neither the path from vertex (1, 0) nor vertex (1, -2) is a better path than the presently recorded path from these vertices. The path from vertex (0, -1) through vertex (1, -1), however, is a better path than the presently recorded path leading directly to Point B. The arrowhead at vertex (0, -1) is therefore changed

from ↑ to → and the Utility value changed from .4889 to .5869. Additionally, adjacent vertices (-1, -1) and (0, -2) are recorded for subsequent examination as Step (8).

(8) Repeat the procedure of Steps (6) and (7) for each of the vertices recorded for subsequent examination in Step (7).

Figure 21 shows the application of Step (8) to our illustrative example. Note that the presently recorded path from each of the re-examined vertices is superior to the path leading through the vertex where the change occurred in the preceeding step, vertex (0, -1). Therefore, no further changes are made, and the algorithm is completed for the first expanded square. Had there been a change required at either re-examined vertex, then an additional step would have been required to re-examine the vertices adjacent to where the change occurred, with additional steps following until no further changes occur.

The algorithm has now determined the best path to Point B from each of the sixteen vertices of the first expanded square, exclusive only of any possible outward excursions. The currently determined "best" paths are shown in Figure 22.

The algorithm is next applied to determine the best path from each vertex on the second expanded square around Point B, encompassing 36 grid squares of the combat engagement area. In this and all subsequent square expansions, the algorithm follows the same sequence as that outlined in the preceding pages for the first expanded square. To wit:

(1) Calculate and record the Utility value, applicable products and direction of each path leading directly inward from vertices of the second expanded square to the corresponding vertices of the first expanded square. These are shown in Figure 23.

(2) Starting at the vertex of highest Utility, move one vertex in a counterclockwise direction and calculate the Utility value of the path leading from that vertex in a clockwise direction. Compare the Utility of the clockwise path with that of the recorded path, select the path of highest utility and retain or change the recorded values and arrowhead direction as appropriate. If a change occurs, note and record for future reference the adjacent vertices that must be checked for recomputed path Utility values and possible direction changes pursuant to the second general rule of the algorithm.

Referring to Figure 23, note that in the case of the illustrative example, the inward path of highest Utility value originates at vertex (2, -3). A clockwise path into this vertex is barred by the impassable grid square adjacent to it in a counterclockwise direction, that is, grid square (20, 10). The next vertex in a counterclockwise direction from which a clockwise path can be calculated is vertex (-3, 1). Accordingly this step of the algorithm, rather than moving one vertex in a counterclockwise direction, will move

to the next counterclockwise vertex from which a clockwise path is possible, vertex (-3, 1). Calculate the Utility value of the clockwise path, and complete the procedure of Step (2) at that vertex. See Figure 24.

(3) Proceeding to the next adjacent vertex in a counterclockwise direction, continue the procedure as with Step (3) of the algorithm sequence applied to the first expanded square until the clockwise path from each vertex of the second expanded square has been checked. See Figure 25.

(4) and (5) Starting again at the same vertex as in Step (2), move one vertex in a clockwise direction and check the Utility value of the path leading from that vertex in a counterclockwise direction. Follow the same procedure as Steps (4) and (5) of the algorithm sequence applied to the first expanded square until the counterclockwise path from each vertex of the second expanded square has been checked. See Figure 26.

At this point, the algorithm has determined the best path to Point B from each of the vertices of the second expanded square, as shown in Figure 27, exclusive of any possible outward excursions (which will be explored at the next larger expanded square), or any possible redirection from the first expanded square resulting from changes in direction and/or Utility values occurring during Steps (2) through (5) above.

(6) Referring to the record of vertices adjacent to those where changes occurred in Steps (2) through (5), calculate the Utility value of the path leading from each recorded vertex to its adjacent, changed vertex. Compare that Utility value with the Utility value of the presently recorded path, select the path of highest utility, and retain or change values and arrowhead direction as appropriate. If a change occurs, note and record the adjacent vertices that must be checked for Utility and possible path redirection as a result of the change.

To illustrate the application of Step (6) to the second expanded square, the vertices where changes occurred in Steps (2) through (5) are indicated in Figure 28 by a small circle at the base of the arrowhead. Referring to Figure 28, the change at vertex (0, -3) requires that the Utility value of the path from adjacent vertex (0, -2) be checked to determine whether the path outward through vertex (0, -3) may not be better than the presently recorded path leading counterclockwise through vertex (1, -2). See Figure 29. Since the Utility value of the outward path calculates to be less than that of the presently recorded, counterclockwise path, the outward path is eliminated, no change occurs at vertex (0, -2) and no further checks are required as a result of the change which occurred at vertex (0, -3). Similarly, as shown in Figure 30, checks at vertices (-1, -2) and (-2, -2) required

by direction changes at vertices (-1, -3) and (-2, -3), respectively, do not disclose a better path than the presently recorded ones and no further checks are required as a result either of the change which occurred at vertex (-1, -3) or the change which occurred at vertex (-2, -3).

Since the checks performed pursuant to Step (6) of the algorithm did not result in any further changes, no additional steps of the algorithm are required at the second expanded square. Figure 31 shows the "best" path from each vertex of the second expanded square, exclusive of any possible excursions to subsequent outer squares.

The algorithm is next applied in the same way to the third, fourth and all-subsequent expanded squares until Point A, the present position of the AML Tank, is encompassed. From that point an additional expanded area is then explored to discover or preclude the possibility that the "best" path may require an initial outward excusion from the vertex of the expanded square encompassing the AML Tank's present position. Figure 31 shows the expansion process successively repeated to encompass the complete area explored. The "best" path as determined by the algorithms is outlined on the sketch of the combat engagement area in Figure 32.

For ease of reference, the algorithm is restated here in its entirety:

SMALLEST SQUARE

(1) Construct a square centered at and comprising the four grid squares immediately surrounding the destination, Point B.

(2) Select one of the center vertices and calculate the Utility of the direct path leading in toward Point B from that vertex. Record the Utility value, the products, ΠDF_T and ΠDF_A, and direction of the path.

(3) Proceeding counterclockwise around Point B, repeat the procedure for each of the remaining three center vertices.

(4) Returning to the originally selected center vertex, move to the adjacent corner vertex in a counterclockwise direction and calculate the Utility of the path from that vertex leading in a clockwise direction. Record the Utility value, the products, ΠDF_T and ΠDF_A, and the direction of the path.

(5) Proceeding to the next adjacent counterclockwise vertex, calculate the Utility of the path from that vertex leading in a clockwise direction. Compare the Utility value of the clockwise path with that of the presently recorded (that is, inward) path, select the path of highest Utility and retain or change the recorded Utility value, products, ΠDF_T and ΠDF_A, and direction, as appropriate. Continue this procedure, moving successively to the next counterclockwise vertex checking the path in a clockwise direction until the Utility value of the clockwise path from each vertex has been calculated. At each corner vertex record the Utility of the

clockwise path together with products ΠDF_T and ΠDF_A, and direction. At each center vertex compare the Utility value of the clockwise path with that of the presently recorded path and retain or change the recorded values and direction, as appropriate.

(6) Returning again to the originally selected center vertex, move to the adjacent corner vertex in a clockwise direction, and calculate the Utility of the path leading from that vertex in a counterclockwise direction. Compare the Utility value of the counterclockwise path with that of the presently recorded path, select the path of highest Utility, and retain or change the recorded Utility value, products, ΠDF_T and ΠDF_A, and direction, as appropriate. Continue the procedure, moving successively to the next clockwise vertex, checking the path in a counterclockwise direction until the counterclockwise path from each vertex has been checked and the recorded Utility value, products and direction at each vertex retained or changed as appropriate to indicate the highest Utility path from that vertex to Point B.

EXPANDED SQUARES

(1) Form an expanded square enlarged by one grid square in each direction surrounding the previous square. Calculate the Utility of each path heading directly inward from the outer vertices of the expanded

square to the corresponding vertices of the previous square. Record the Utility values, the products, ΠDF_T and ΠDF_A, and the path direction.

(2) Starting at the vertex of highest Utility, move one vertex in a counterclockwise direction and calculate the Utility value of the path leading from that vertex in a clockwise direction. Compare the Utility of the clockwise path with that of the recorded path,* select the path of highest Utility, and retain or change the recorded Utility value, products, ΠDF_T and ΠDF_A, and direction, as appropriate. If a change occurs, note and record for future reference the adjacent vertices that must be checked for recomputed path Utility values and possible direction changes pursuant to the second general rule of the algorithm.**

(3) Proceeding to the next adjacent vertex in a counterclockwise direction, calculate the Utility value of the path leading from that vertex in a clockwise direction. Compare and select the path of highest Utility as in Step (2) above, noting and recording

---

* If there is no recorded path, as will be the case at the outer corner vertices, then record the Utility value, products, ΠDF_T and ΠDF_A, and direction of the clockwise path and go on to Step (3).

**Rule 2. Any vertex at which a change in either Utility value or direction has occurred must have the immediately surrounding vertices (exclusive of the vertex to which the changed path now leads since that direction has been checked) checked to see whether or not those surrounding vertices now require a direction and/or Utility value change.

for future reference vertices where changes occur together with their potentially affected adjacent vertices which must be subsequently checked in accordance with the second general rule of the algorithm. Continue this procedure moving successively to each next counterclockwise vertex, checking the path in a clockwise direction, until the clockwise path from each vertex has been checked.

(4) Starting again at the same vertex as in Step (2), move one vertex in a clockwise direction to check the counterclockwise path. If the direction of the presently recorded path from the next counterclockwise vertex is clockwise, then the counterclockwise path is eliminated from consideration. (The first general rule of the algorithm applies.***) If the path from the next adjacent counterclockwise vertex does not lead in a clockwise direction, then calculate the Utility value of the counterclockwise path, compare that Utility value with the Utility value of the presently recorded path, select the path of highest Utility, and retain or change the recorded

***Rule 1: A path from any vertex that would lead in a direction against the recorded direction of a next adjacent vertex can be eliminated a priori.

values and direction, as appropriate. Again, pursuant to the second general rule of the algorithm, whenever changes occur, note and record the potentially affected adjacent vertices to be subsequently checked.

(5) Proceeding to the next adjacent vertex in a clockwise direction, repeat the procedure of Step (4) above, moving successively to each next clockwise vertex, checking the path in a counterclockwise direction, until the counterclockwise path from each vertex has been checked.

(6) Recall the record of vertices to be re-checked pursuant to the second general rule of the algorithm by virtue of their adjacency to vertices where changes occurred during Steps (2) through (5). Calculate the Utility of the path leading from each recorded vertex to its adjacent, changed vertex. Compare that Utility value with the Utility value of the presently recorded path, select the path of highest Utility, and retain or change the recorded values and direction as appropriate. If a change occurs, note and record the additional adjacent vertices that must be checked for Utility and possible path redirection as a result of the change.

(7) through (n)

Recall the record of vertices to be re-checked because of changes occurring during the preceeding

step and repeat the procedure of the preceeding step to determine whether any additional checks are required. When no additional checks are required, the algorithm is completed for the expanded square formed in Step (1) above.

The sequence of steps for the expanded square just completed is then repeated for each successively larger expanded square until the grid square containing the present position of the AML Tank has been encompassed and the "best" path from the present position to the destination has been determined.

In order to test the efficacy of the algorithm, it has been manually applied to the combat engagement situation depicted in Figure 1. The maze of path calculations, comparisons and changes involved are apparent from examination of Figures 3 through 31. Although an extremely cumbersone exercise conducted manually, it can readily be accomplished by a computer. The manual test, however, provided an opportunity for discrete scrutiny of the algorithm at each of the individual steps and for observing the sensitivity to features of terrain and other pertinent factors affecting defensive and offensive posture. The algorithm proved remarkably adept in evaluative path selection throughout the area explored. It is interesting to observe, for example, some of the path selections made from points (vertices) of the combat engagement area in addition to the final path determination from Point A.

Referring to Figure 33, note that at the first expanded square, had the present position of the AML Tank been at vertex (-2, 2), the selected path would be first to gain cover behind the rise in the terrain between grid squares (16, 13) and (16, 14) and then proceed to Point B. Similarly, had the AML Tank been at vertex (-2, -2) or (-2, -1), it would have sought the cover from the same rise in terrain while keeping the Trainee Tank within range of its own main armament. An AML Tank at vertex (0, -2), on the other hand, outside the main armament range of both tanks would seek the cover and concealment of the river bank en route to Point B.

As the area of exploration is expanded outward from Point B, circuitous paths seeking the cover and concealment of the river bank are increasingly favored over direct or exposed paths. From vertices (-3, -3), (-4, -3) and (-5, -3), for example, the initial move is to gain the cover of the riverbank and then turn to follow the riverbank to Point B. See Figure 34. Influencing factors in these path selections, in addition to defensive cover and concealment afforded by the riverbank and its overhanging trees, are the relative positions of the AML and Trainee Tanks, and the higher importance weight given to "survivability" as opposed to "attackability" by the Valuated State Space of the AML Tank Commander's Purpose. The fact that the AML Tank would be required to present its vulnerable aspect to the Trainee Tank as it approached Point B was also an influencing consideration. From vertices (-5, -2), (-4, -2) and (-3, 2) the closer

proximity of the two tanks and greater distance to a covered path influenced the selection in favor of paths that maximize the "attackability" of the AML Tank thus reducing the defensive posture of the Trainee Tank. In other words, the algorithm, in effect, determined that the AML Tank Commander could best achieve his purpose by "shooting it out" with the Trainee Tank en route to Point B rather than first attempting to improve his own defensive posture. It should be noted here, however, that had the AML Tank's present position actually been at one of the vertices (-5, -2), (-4, -2) or (-3, -2), the AML Program would in all probability have selected some destination point other than Point B. These examples are cited only to point out the logical basis for the evaluative path selection generated by the algorithm.

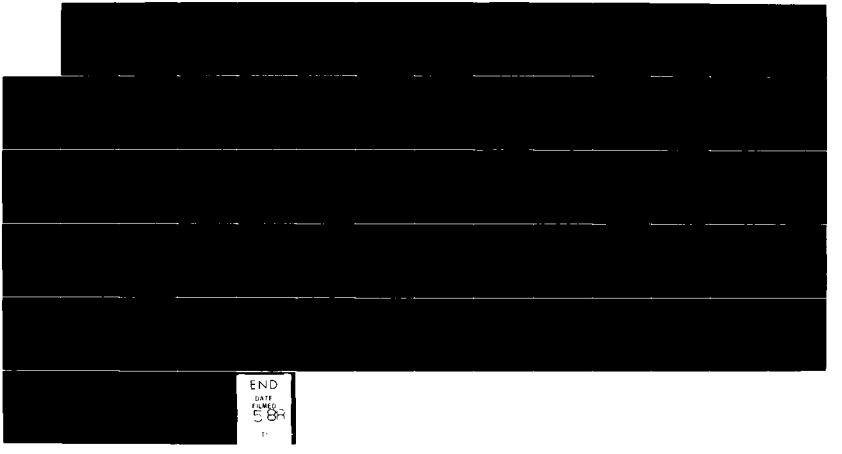
Another interesting path decision point which demonstrates the sensitivity of the algorithm is shown in Figure 35. Note the small hill in the vicinity of grid squares (11, 6) and (12, 6). A particularly vulnerable position for the AML Tank would be the crest of this hill where the AML Tank would be silhouetted against the sky and just within lethal range of the Trainee Tank's main armament. The algorithm appropriately selects paths to avoid that hazard. Note that from vertices (-8, -6) and (-8, -7) the selected path skrits around behind the hill, both to avoid cresting the hill and to gain cover behind the hill. From vertices (-7, -6) and (-6, -6), however, the most direct path to the cover of the riverbank is selected.

The final path, selected by the algorithm as the "best" path from Point A to Point B, is shown in Figure 32.

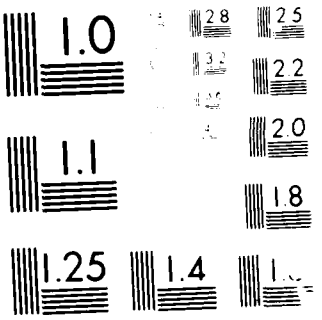
As noted earlier in the Disucssion, a major effort of the quarter just completed has been directed toward developing and testing the algorithm for path selection as described in the preceeding pages. The balance of the contract effort will be concentrated on refining the valuated state space and creating the architecture, flow diagrams and initial software specifications for the AML Program.

AD-A191 923 THE ADAPTIVE MANEUVERING LOGIC IN TANK WARFARE SIMULATION(U) DECISION SCIENCE INC SAN DIEGO CA A J QUENS ET AL. 20 MAY 82 DST-82-413-F 2/2

UNCLASSIFIED MDA903-81-C-0589 F/G 12/5 NL



END
DATE FILMED
5 83


1.0
1.1
1.25
1.4
2.8
3.2
2.5
2.2
2.0
1.8

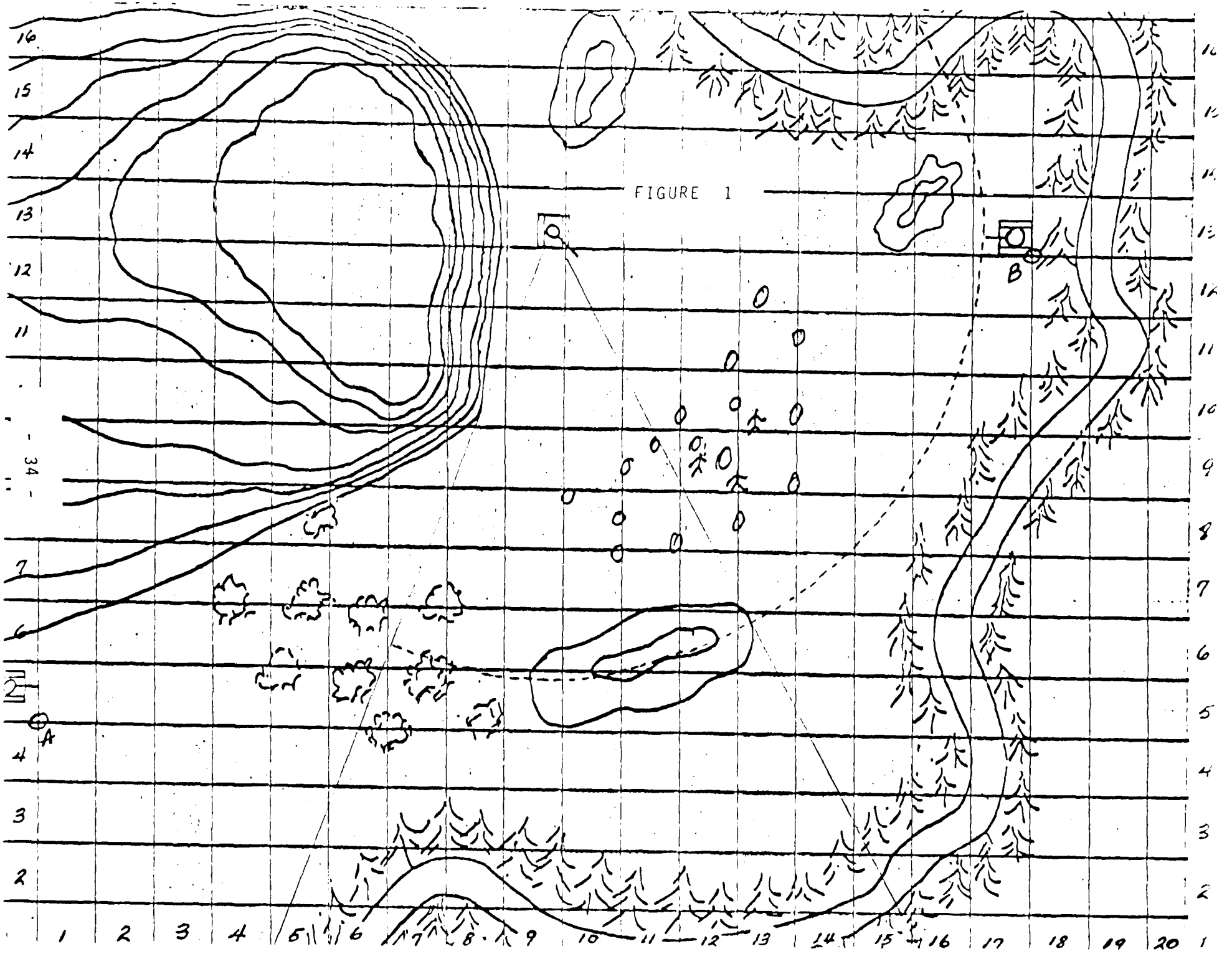


FIGURE 1

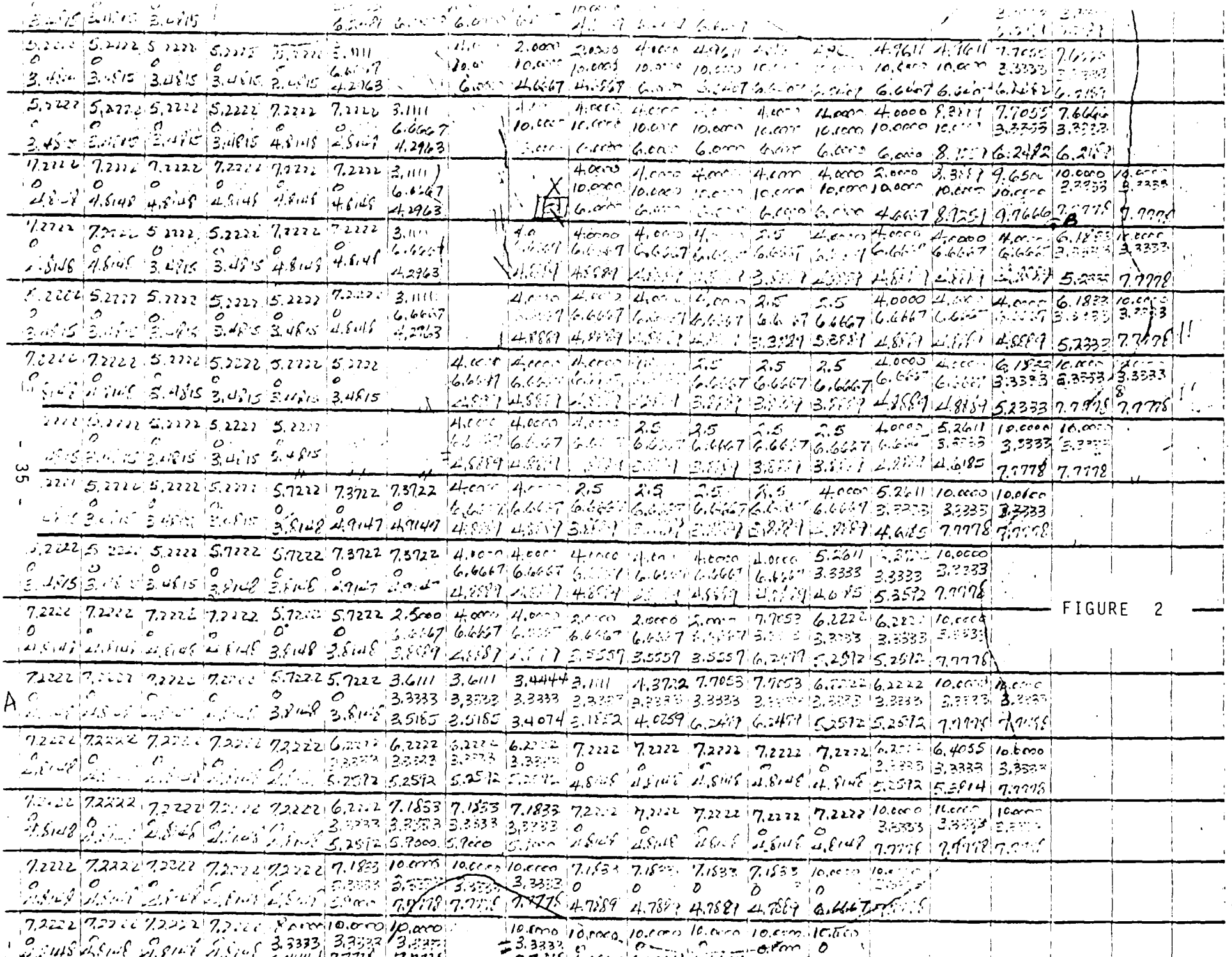


FIGURE 2

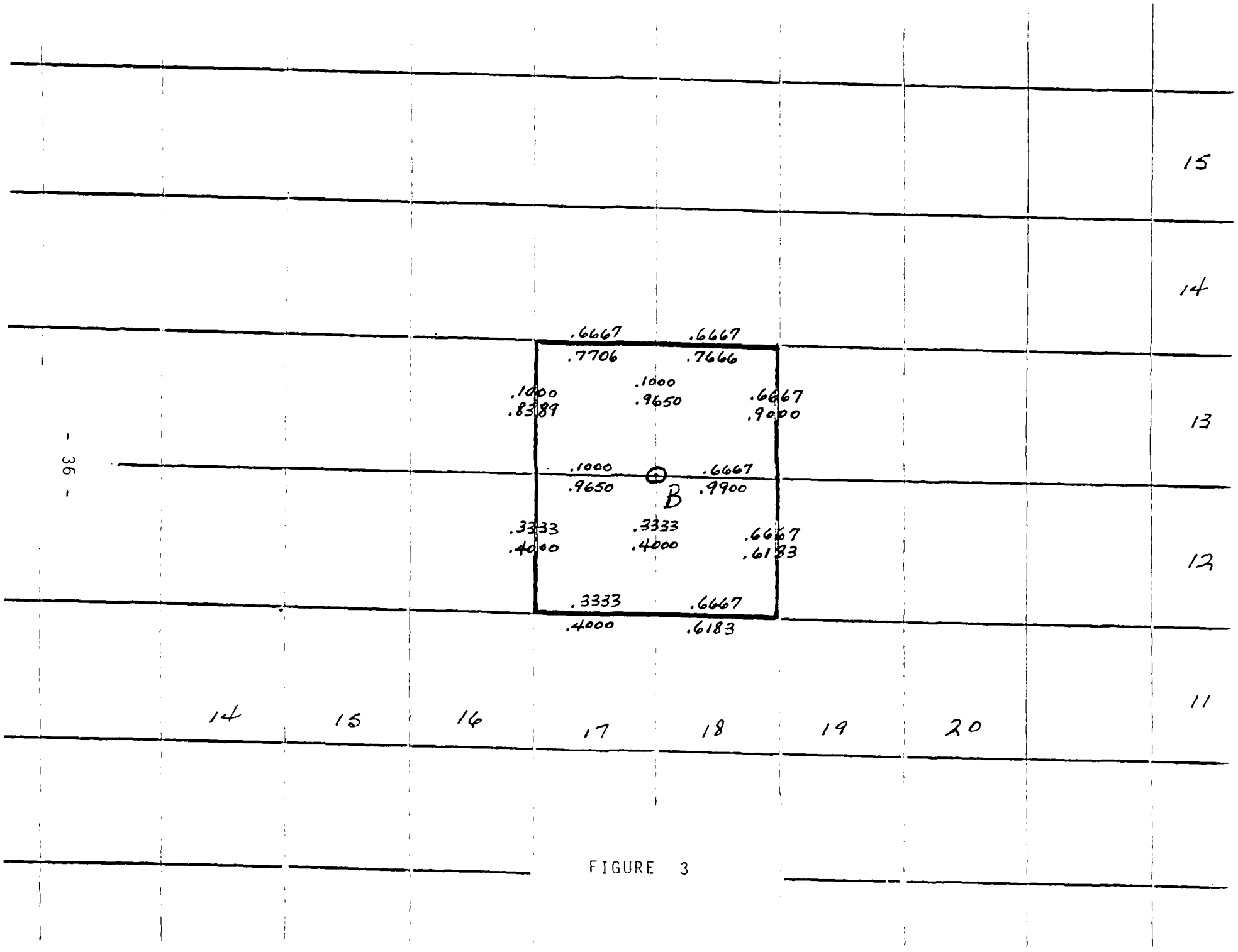

15
14
13
12
11
.6667
.7706
.6667
.7666
.1000
.9650
.1000
.8389
.6667
.9000
.1000
.9650
.6667
.9900
B
.3333
.4000
.3333
.4000
.6667
.6183
.3333
.4000
.6667
.6183
14
15
16
17
18
19
20
- 36 -

FIGURE 3

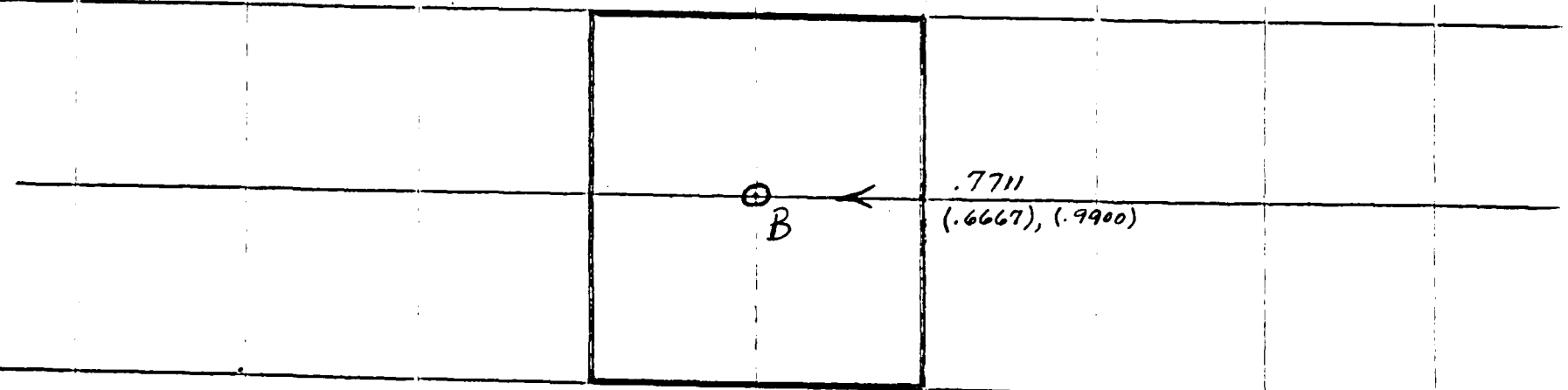


FIGURE 4

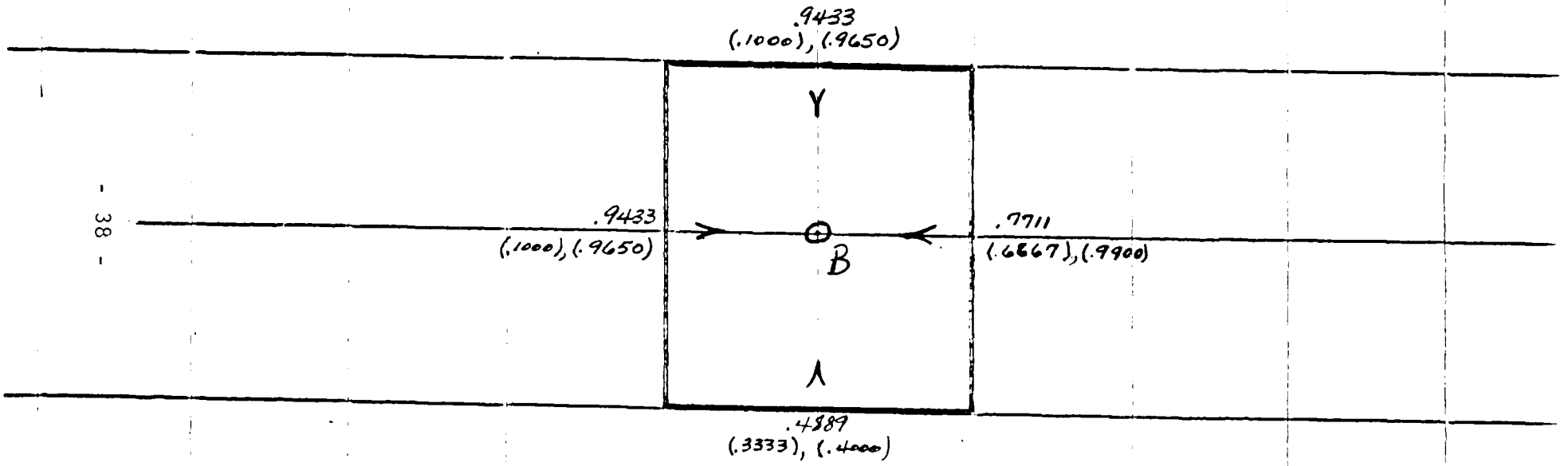

.9433
(.1000), (.9650)
Y
.9433
(.1000), (.9650)
B
.7711
(.6667), (.9900)
A
.4889
(.3333), (.4000)

FIGURE 5

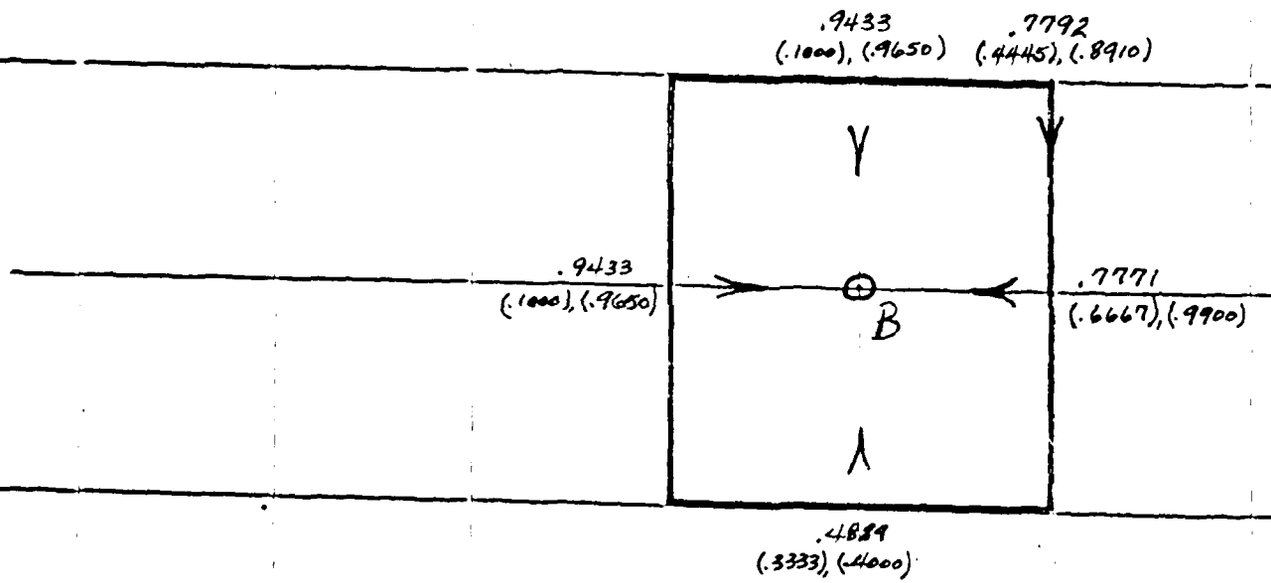

.9433
(.1000), (.9650)
.7792
(.4445), (.8910)
.9433
(.1000), (.9650)
B
.7771
(.6667), (.9900)
.4889
(.3333), (.4000)

FIGURE 6

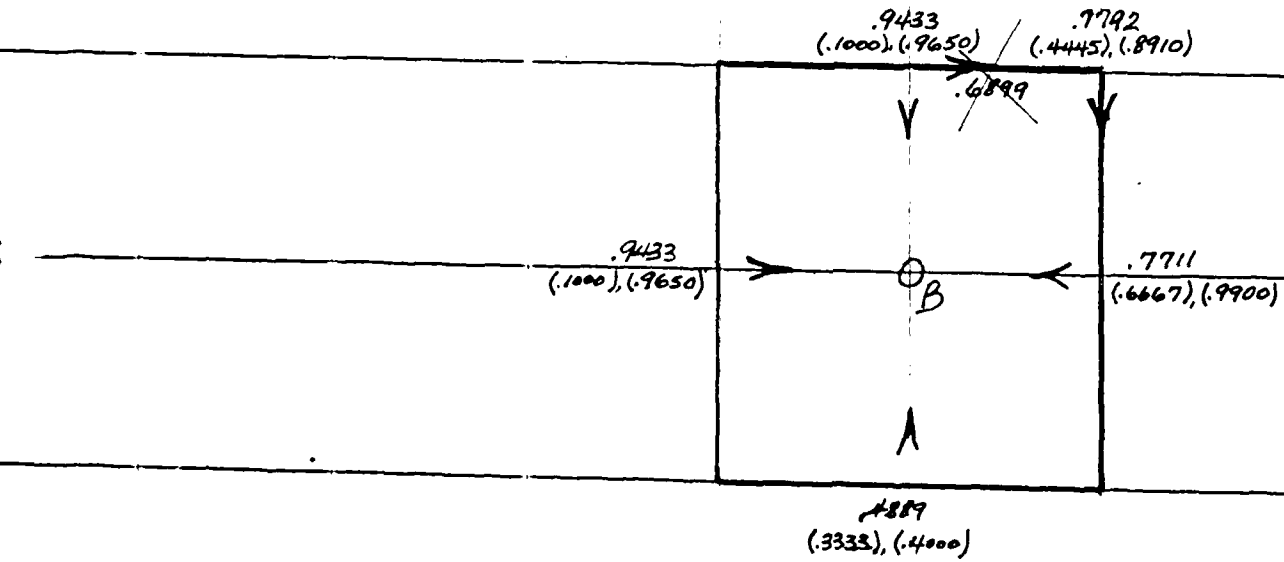

.9433
(.1000), (.9650)
.7792
(.4445), (.8910)
.6899
.9433
(.1000), (.9650)
B
.7711
(.6667), (.9900)
.4889
(.3333), (.4000)

FIGURE 7

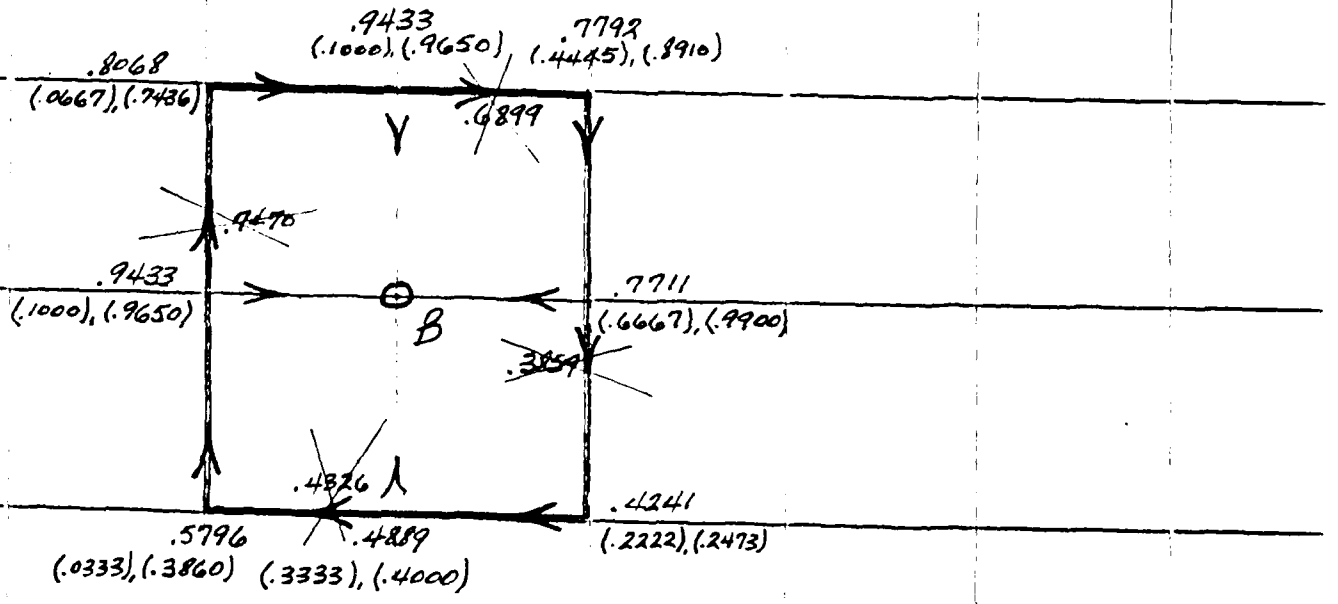

.9433
(.1000), (.9650)
.7792
(.4445), (.8910)
.8068
(.0667), (.7486)
.6899
.9470
.9433
(.1000), (.9650)
B
.7711
(.6667), (.9900)
.3854
.4826
.4241
(.2222), (.2473)
.5796
(.0333), (.3860)
.4889
(.3333), (.4000)

FIGURE 8

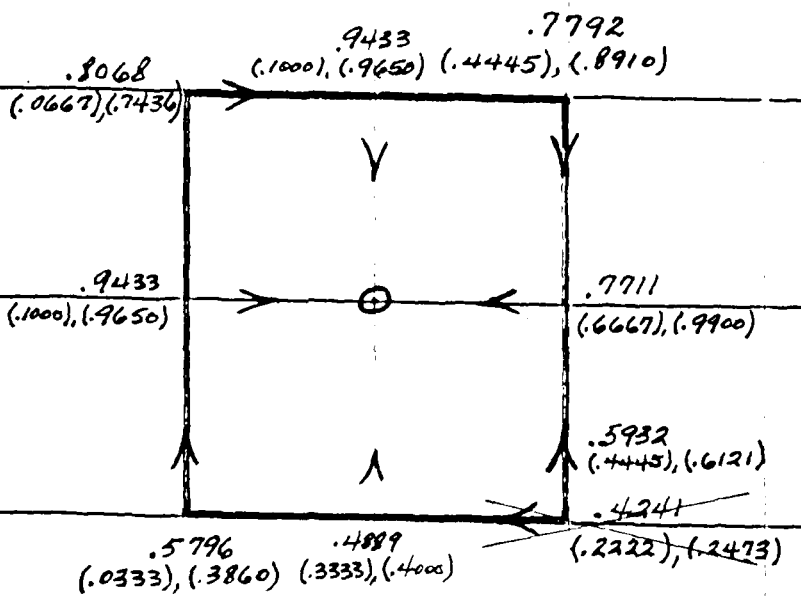

.8068
(.0667), (.7436)
.9433
(.1000), (.9650)
.7792
(.4445), (.8910)
.9433
(.1000), (.9650)
.7711
(.6667), (.9900)
.5932
(.4445), (.6121)
.4241
(.2222), (.2473)
.5796
(.0333), (.3860)
.4889
(.3333), (.4000)

FIGURE 9

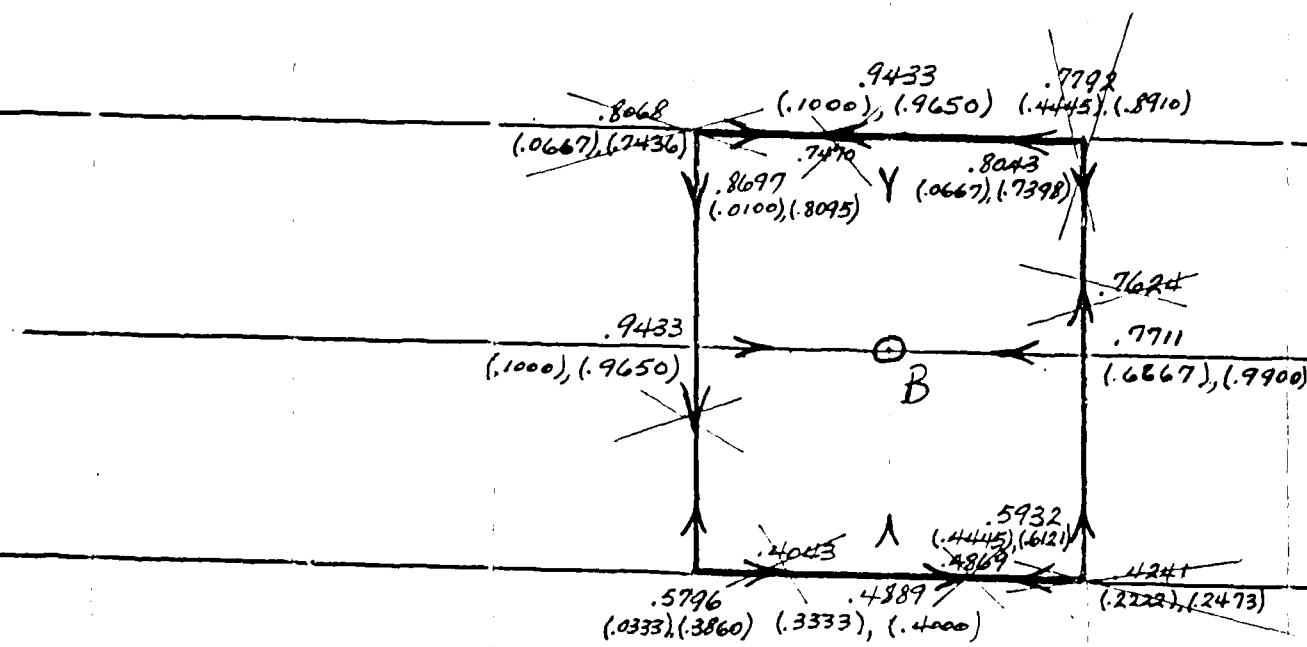

.9433
(.1000), (.9650)
.7792
(.4445), (.8910)
.8068
(.0667), (.2436)
.7470
.8043
(.0667), (.7398)
.8697
(.0100), (.8095)
.7624
.9433
(.1000), (.9650)
B
.7711
(.6667), (.9900)
.5932
(.4445), (.6121)
.4869
.5796
(.0333), (.3860)
.4889
(.3333), (.4000)
(.2222), (.2473)

FIGURE 10

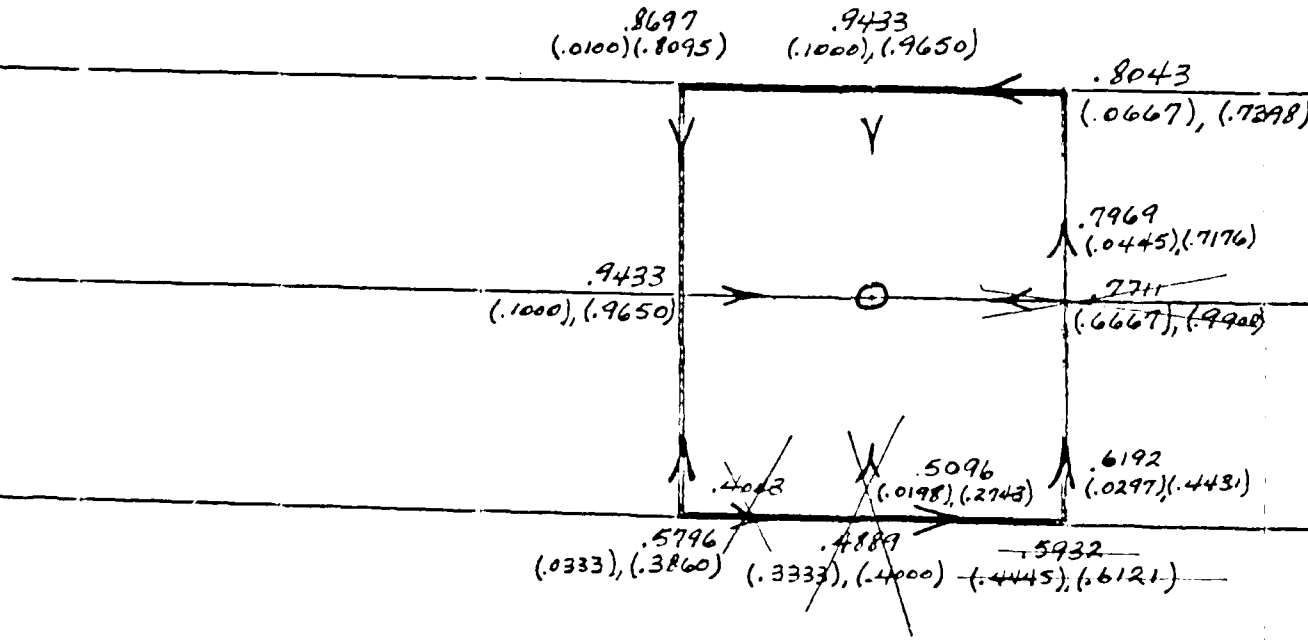

.8697
(.0100)(.8095)
.9433
(.1000), (.9650)
.8043
(.0667), (.7898)
.7969
(.0445), (.7176)
.9433
(.1000), (.9650)
.2744
(.6667), (.9908)
.5096
(.0198), (.2743)
.6192
(.0297), (.4431)
.5796
(.0333), (.3860)
.4889
(.3333), (.4000)
.5932
(.4445), (.6121)

FIGURE 11

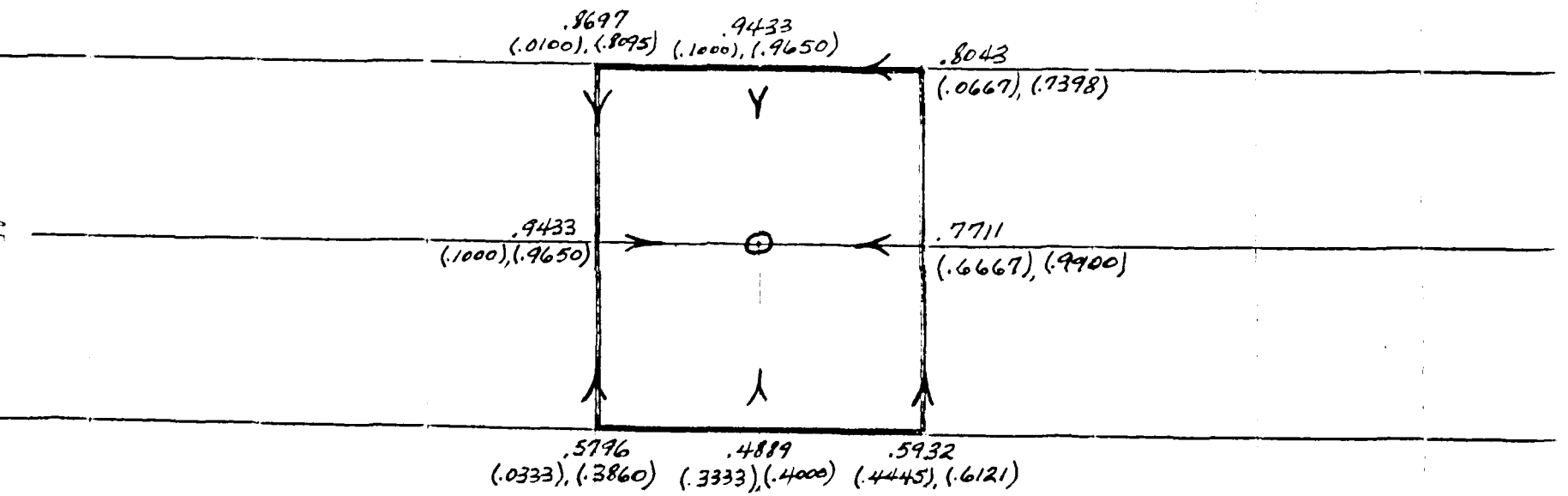

.8697
(.0100), (.8095)
.9433
(.1000), (.9650)
.8043
(.0667), (.7398)
.9433
(.1000), (.9650)
.7711
(.6667), (.9900)
.5796
(.0333), (.3860)
.4889
(.3333), (.4000)
.5932
(.4445), (.6121)

FIGURE 12

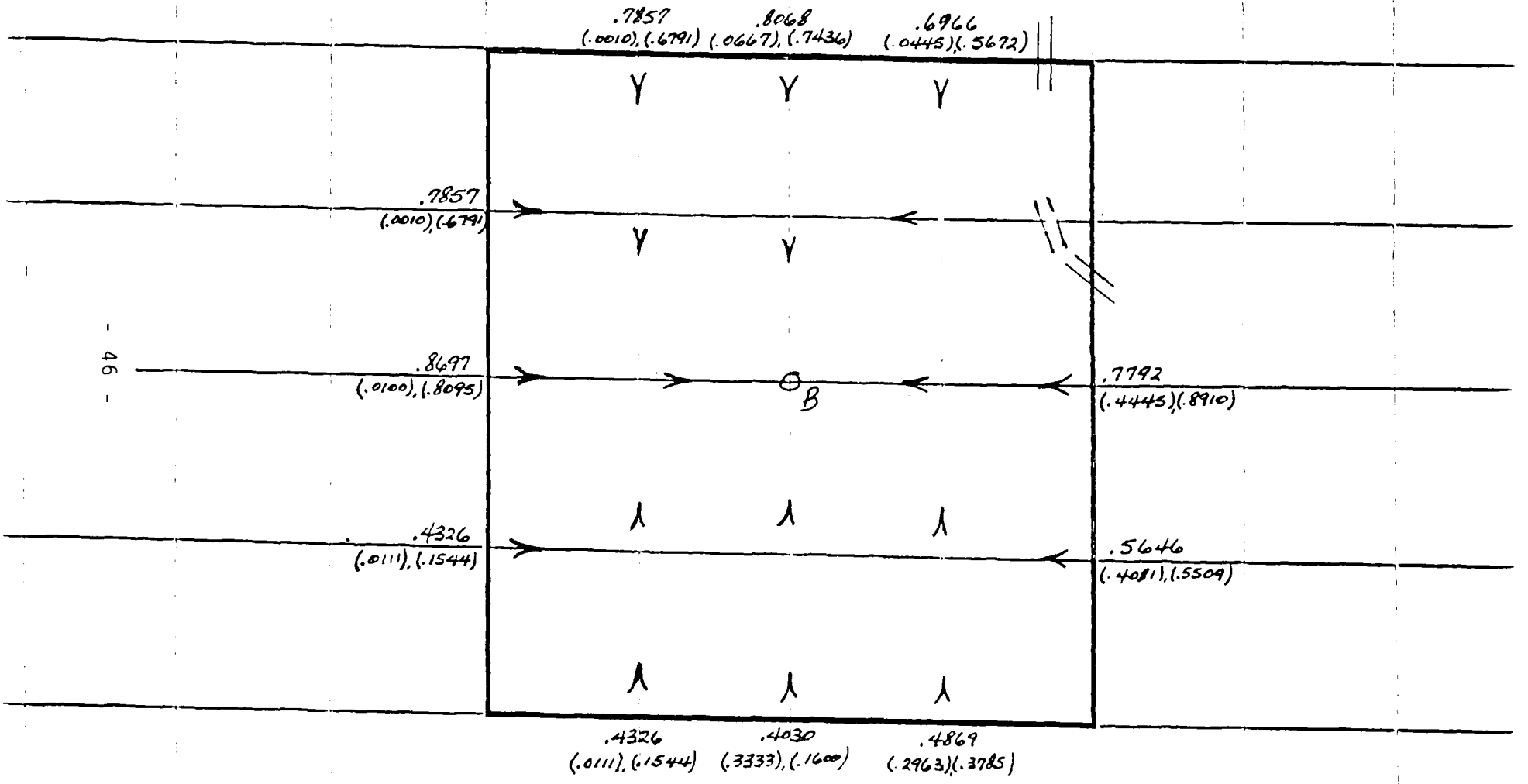

.7857
(.0010), (.6791)
.8068
(.0667), (.7436)
.6966
(.0445), (.5672)
.7857
(.0010), (.6791)
.8697
(.0100), (.8095)
B
.7792
(.4445), (.8910)
.4326
(.0111), (.1544)
.5646
(.4081), (.5509)
.4326
(.0111), (.1544)
.4030
(.3333), (.1600)
.4869
(.2963), (.3785)

FIGURE 13

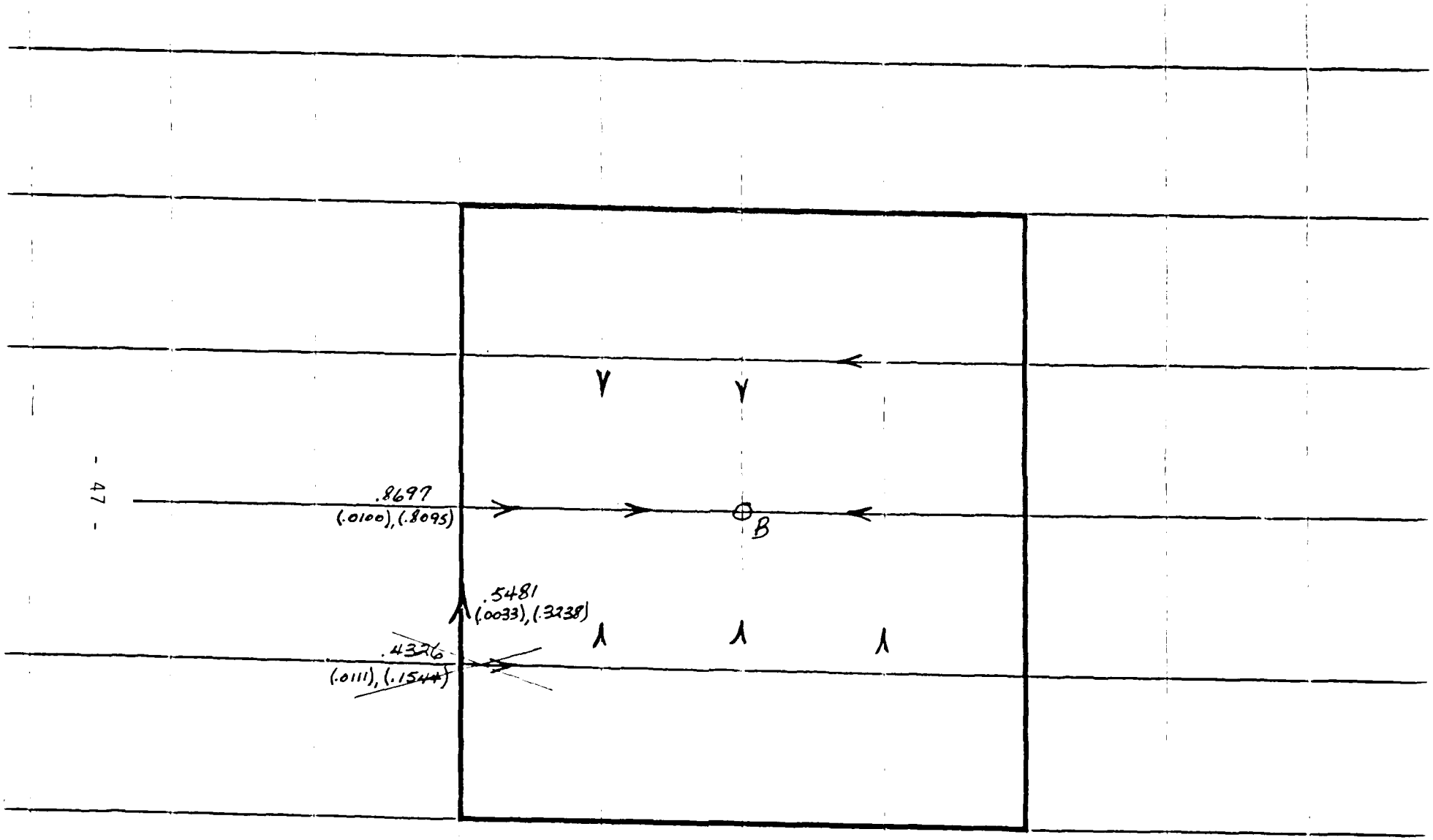


FIGURE 14

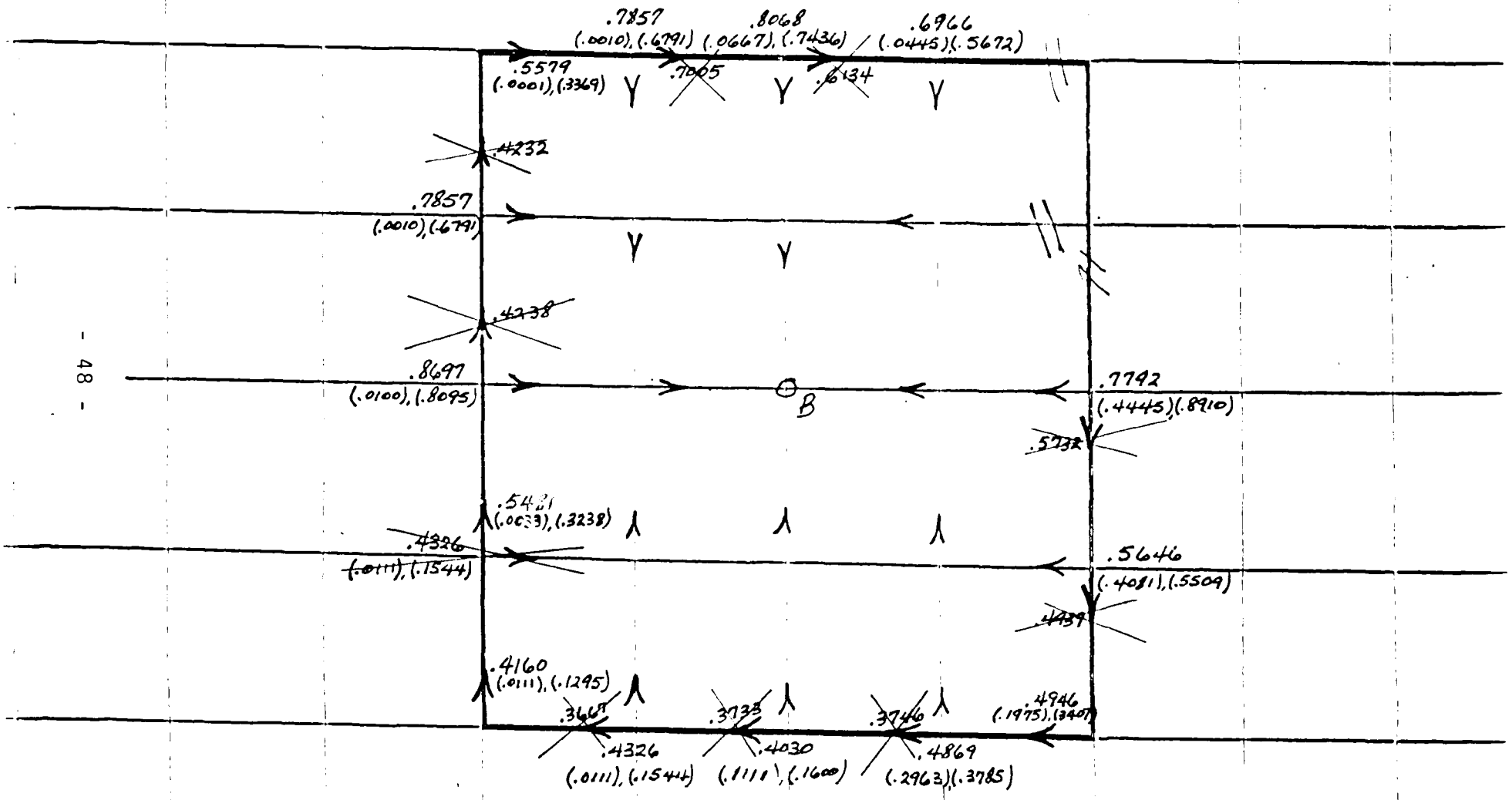

.7857
(.0010), (.6791)
.8068
(.0667), (.7436)
.6966
(.0445), (.5672)
.5579
(.0001), (.3369)
.7005
.6134
.4232
.7857
(.0010), (.6791)
.4238
.8697
(.0100), (.8095)
B
.7792
(.4445), (.8910)
.5732
.5421
(.0033), (.3238)
.4326
(.0111), (.1544)
.5646
(.4081), (.5509)
.4439
.4160
(.0111), (.1295)
.3669
.3738
.3746
.4946
(.1975), (.3401)
.4326
(.0111), (.1544)
.4030
(.1111), (.1600)
.4869
(.2963), (.3785)

FIGURE 15

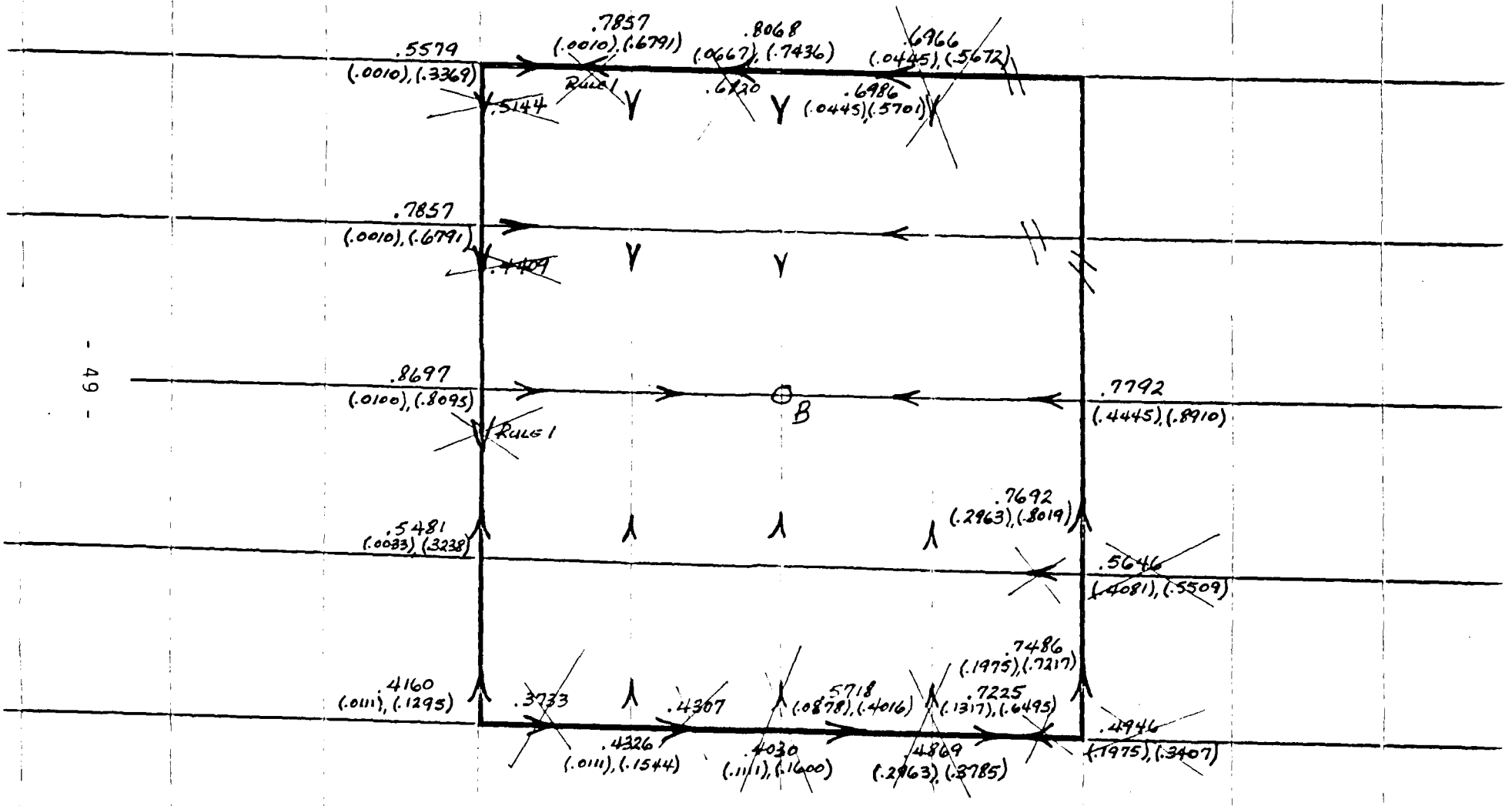

.5579
(.0010), (.3369)
.5144
.7857
(.0010), (.6791)
Rule 1
.7857
(.0010), (.6791)
.8068
(.0667), (.7436)
.6720
.6966
(.0445), (.5672)
.6986
(.0445), (.5701)
.8697
(.0100), (.8095)
Rule 1
B
.7792
(.4445), (.8910)
.5481
(.0083), (.3238)
.7692
(.2963), (.8019)
.5646
(.4081), (.5509)
.4160
(.0111), (.1295)
.3733
.4326
(.0111), (.1544)
.4367
.4030
(.1111), (.1600)
.5718
(.0878), (.4016)
.4869
(.2963), (.3785)
.7486
(.1975), (.7217)
.7225
(.1317), (.6495)
.4946
(.1975), (.3407)

FIGURE 16

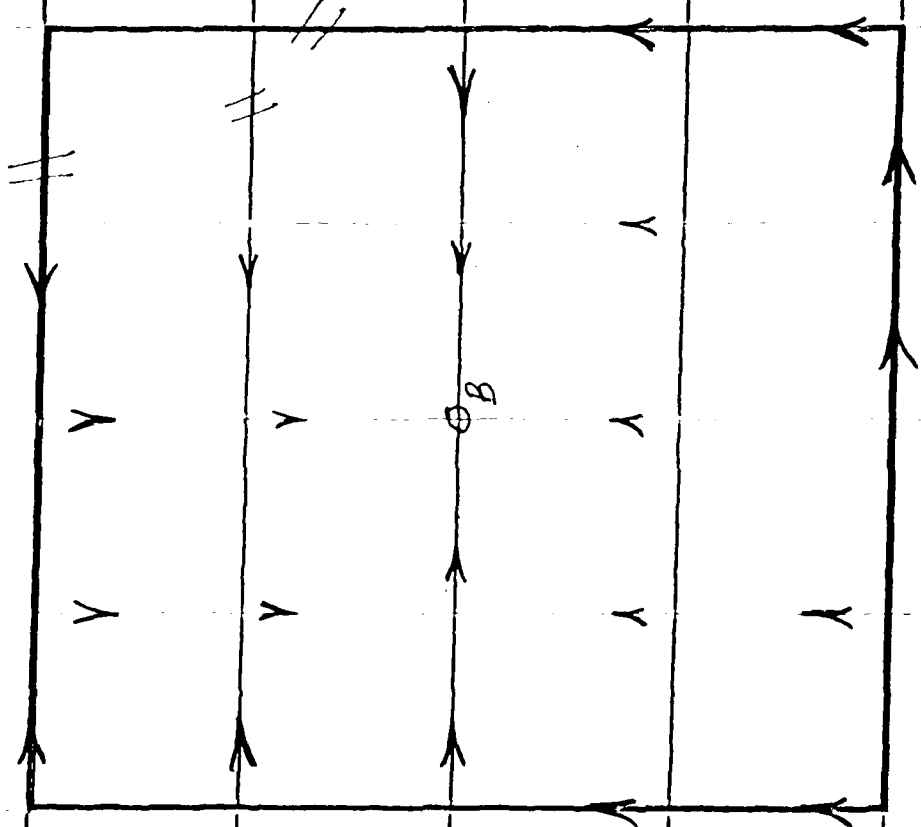


FIGURE 17

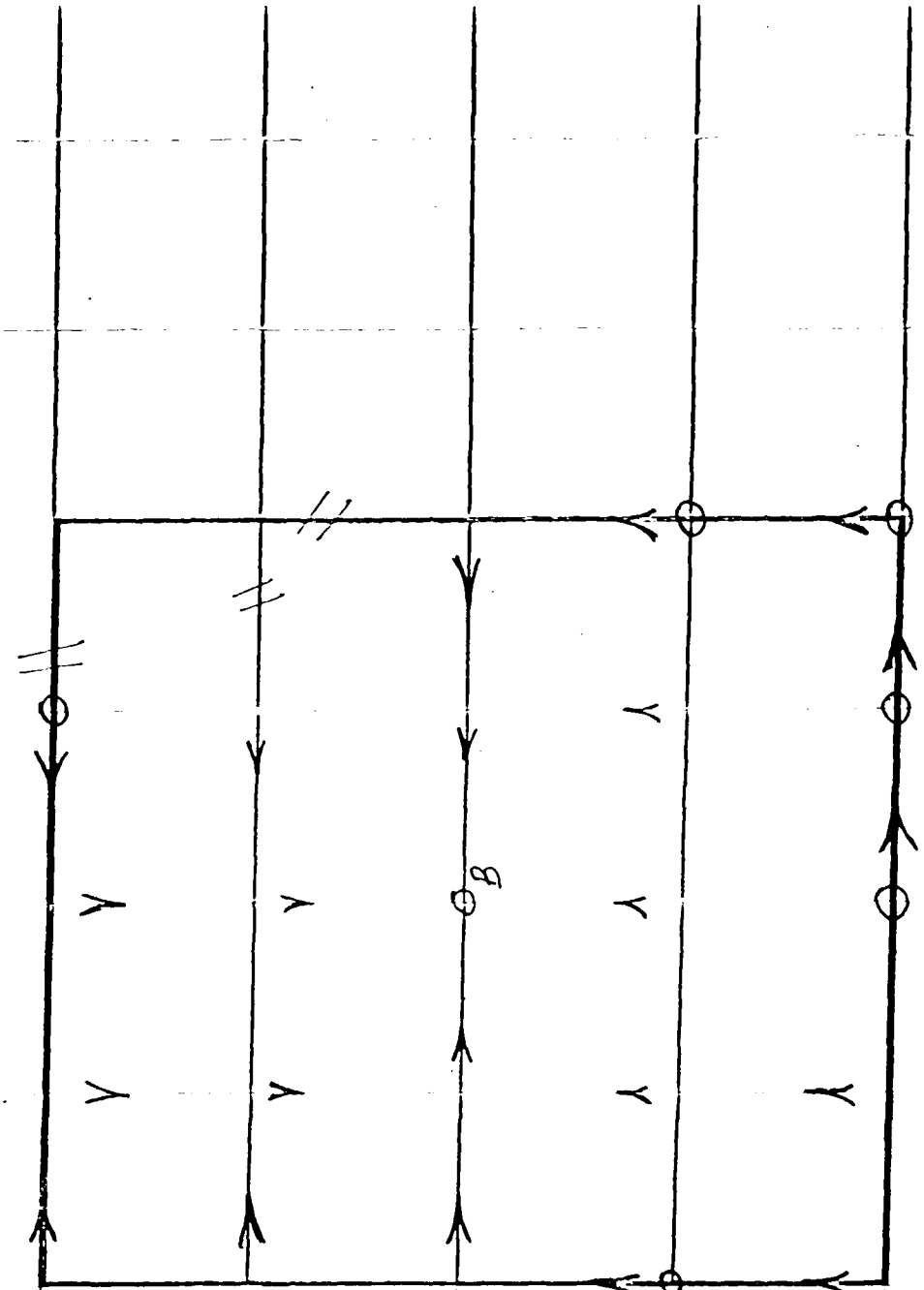


FIGURE 18

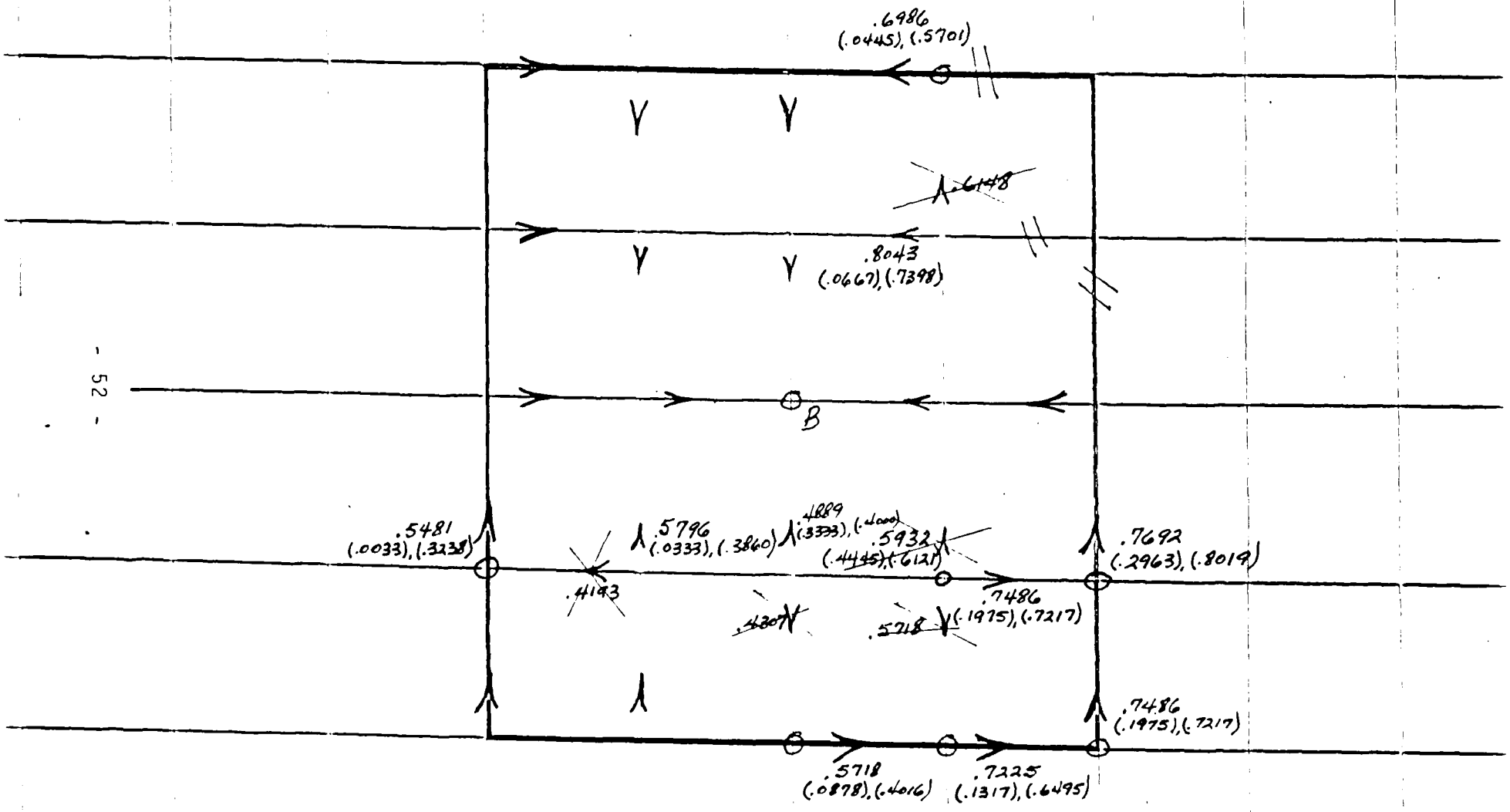

.6986
(.0445), (.5701)
.6148
.8043
(.0667), (.7398)
B
.5481
(.0033), (.3238)
.5796
(.0333), (.3860)
.4889
(.3333), (.4000)
.5932
(.4445), (.6121)
.7692
(.2963), (.8019)
.4193
.5218
.7486
(.1975), (.7217)
.7486
(.1975), (.7217)
.5718
(.0878), (.4016)
.7225
(.1317), (.6495)

FIGURE 19

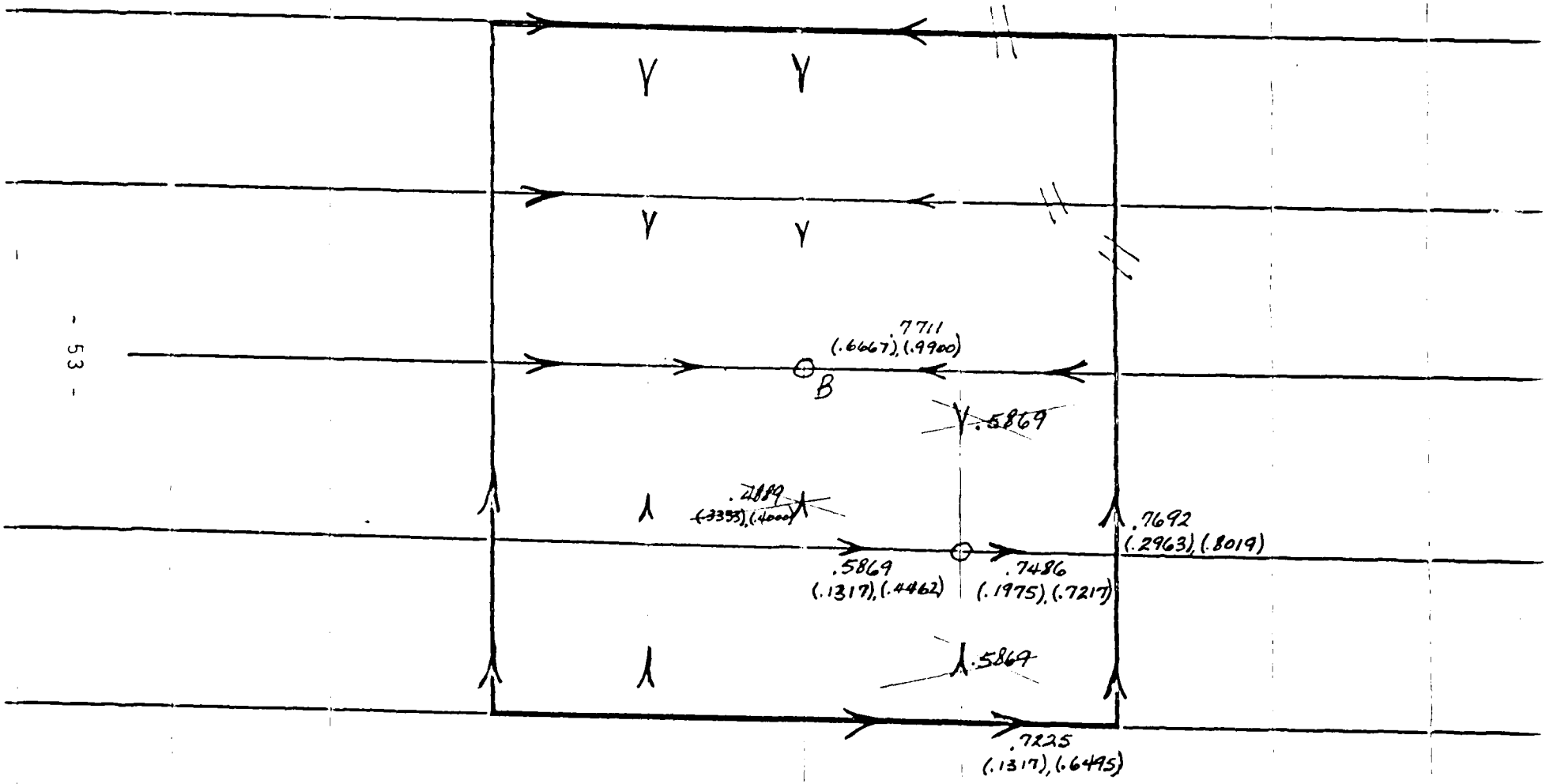

.7711
(.6667), (.9900)
B
.5869
.2889
(.3395), (.4000)
.7692
(.2963), (.8019)
.5869
(.1317), (.4462)
.7486
(.1975), (.7217)
.5869
.7225
(.1317), (.6495)

FIGURE 20

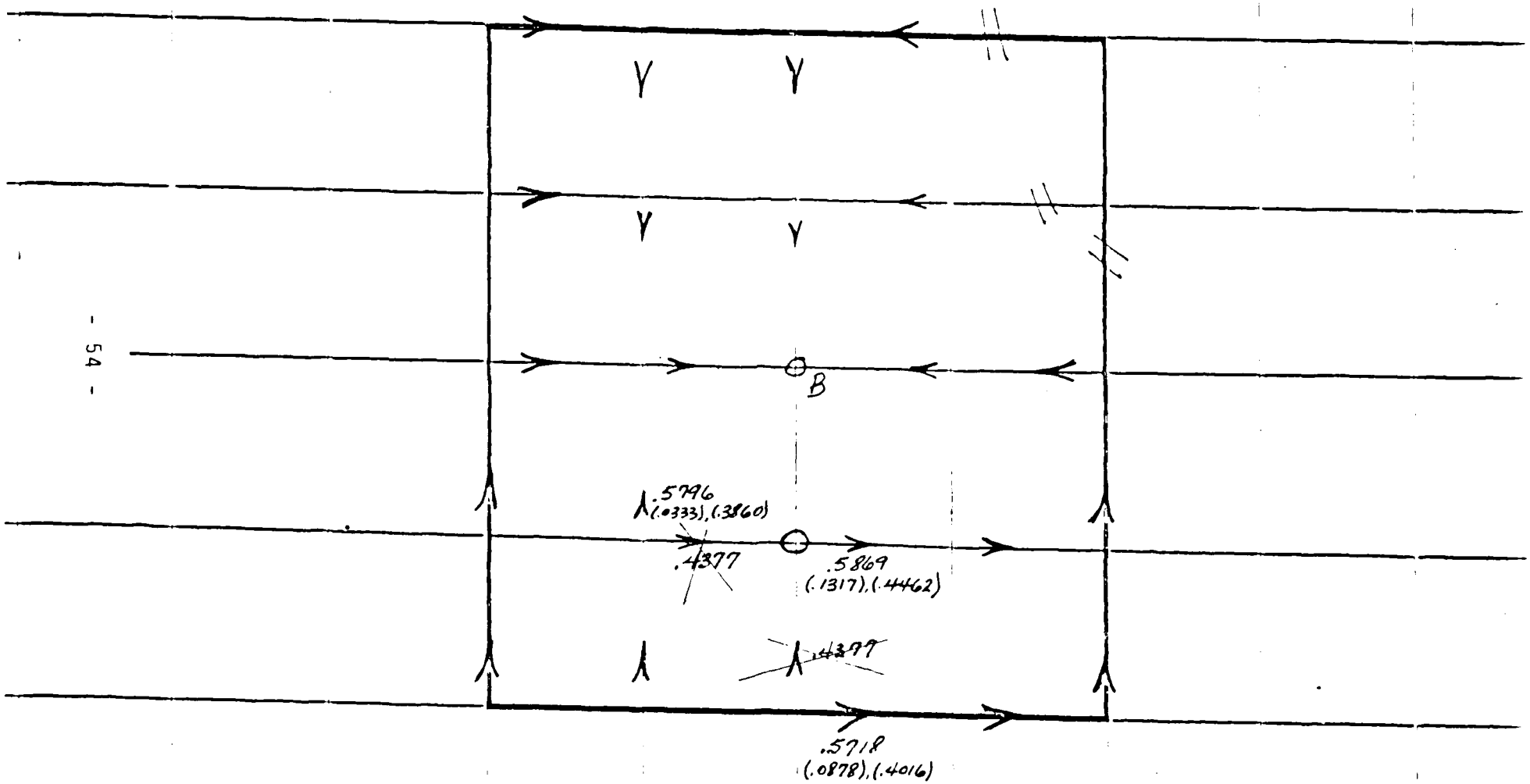


FIGURE 21

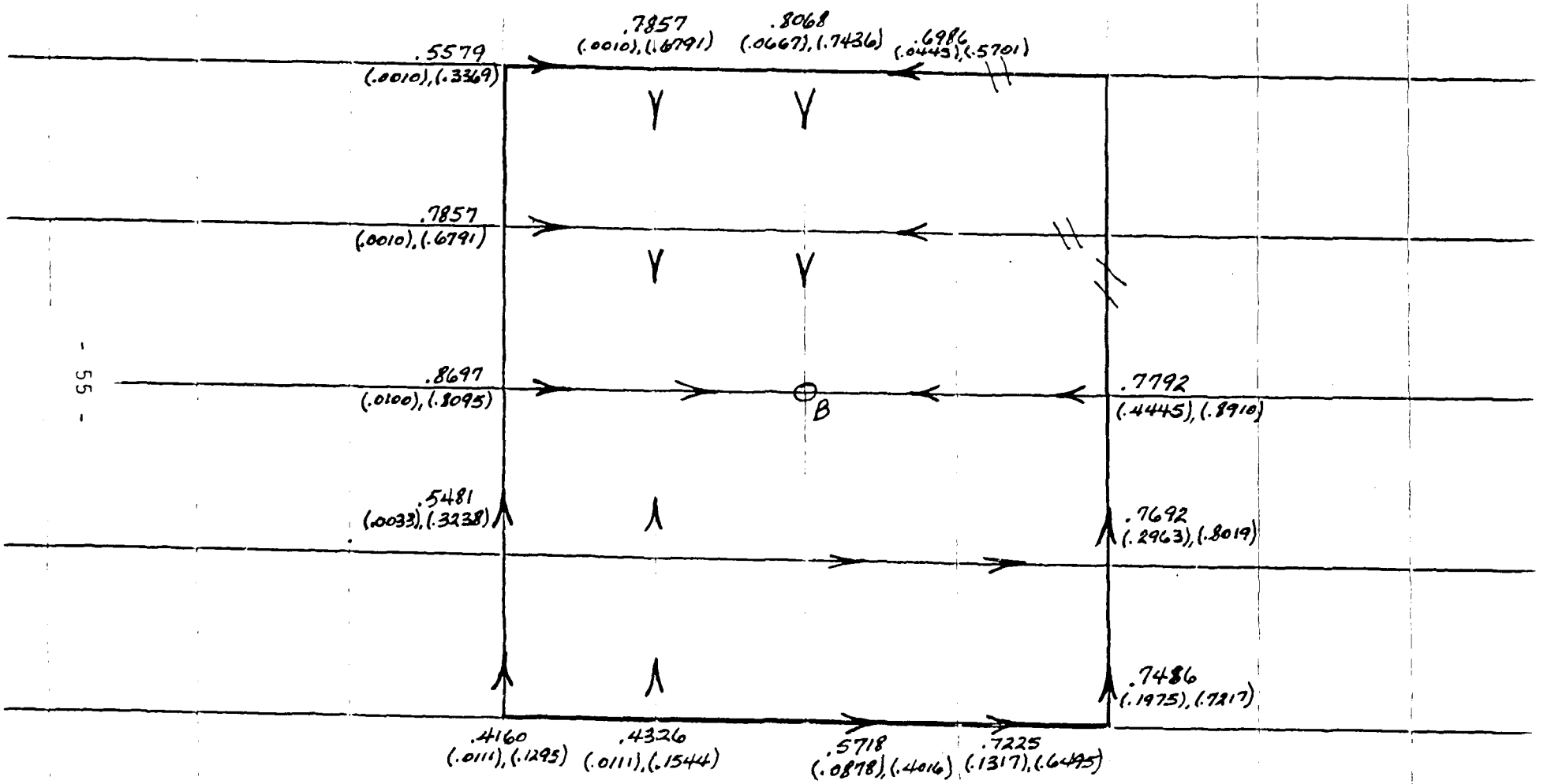

.5579
(.0010), (.3369)
.7857
(.0010), (.6791)
.8068
(.0667), (.7436)
.6986
(.0443), (.5701)
.7857
(.0010), (.6791)
.8697
(.0100), (.8095)
B
.7792
(.4445), (.8910)
.5481
(.0033), (.3238)
.7692
(.2963), (.8019)
.7486
(.1975), (.7217)
.4160
(.0111), (.1295)
.4326
(.0111), (.1544)
.5718
(.0878), (.4016)
.7225
(.1317), (.6495)

FIGURE 22

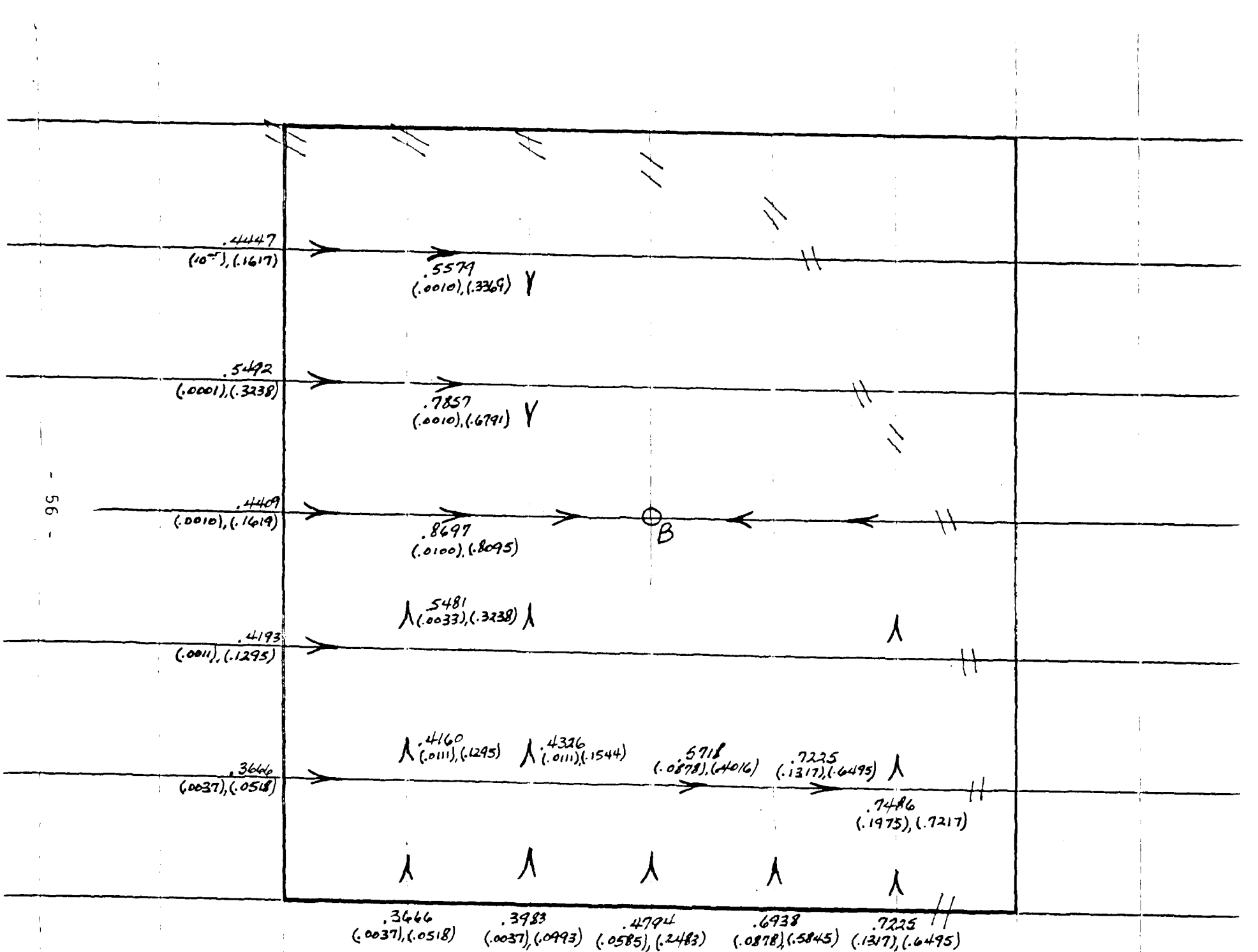

.4447
(10⁻⁵), (.1617)
.5579
(.0010), (.3369)
.5492
(.0001), (.3238)
.7857
(.0010), (.6791)
.4409
(.0010), (.1619)
.8697
(.0100), (.8095)
B
.5481
(.0033), (.3238)
.4193
(.0011), (.1295)
.4160
(.0111), (.1295)
.4326
(.0111), (.1544)
.5718
(.0878), (.4016)
.7225
(.1317), (.6495)
.3646
(.0037), (.0518)
.7486
(.1975), (.7217)
.3646
(.0037), (.0518)
.3983
(.0037), (.0493)
.4794
(.0585), (.2483)
.6938
(.0878), (.5845)
.7225
(.1317), (.6495)

FIGURE 23

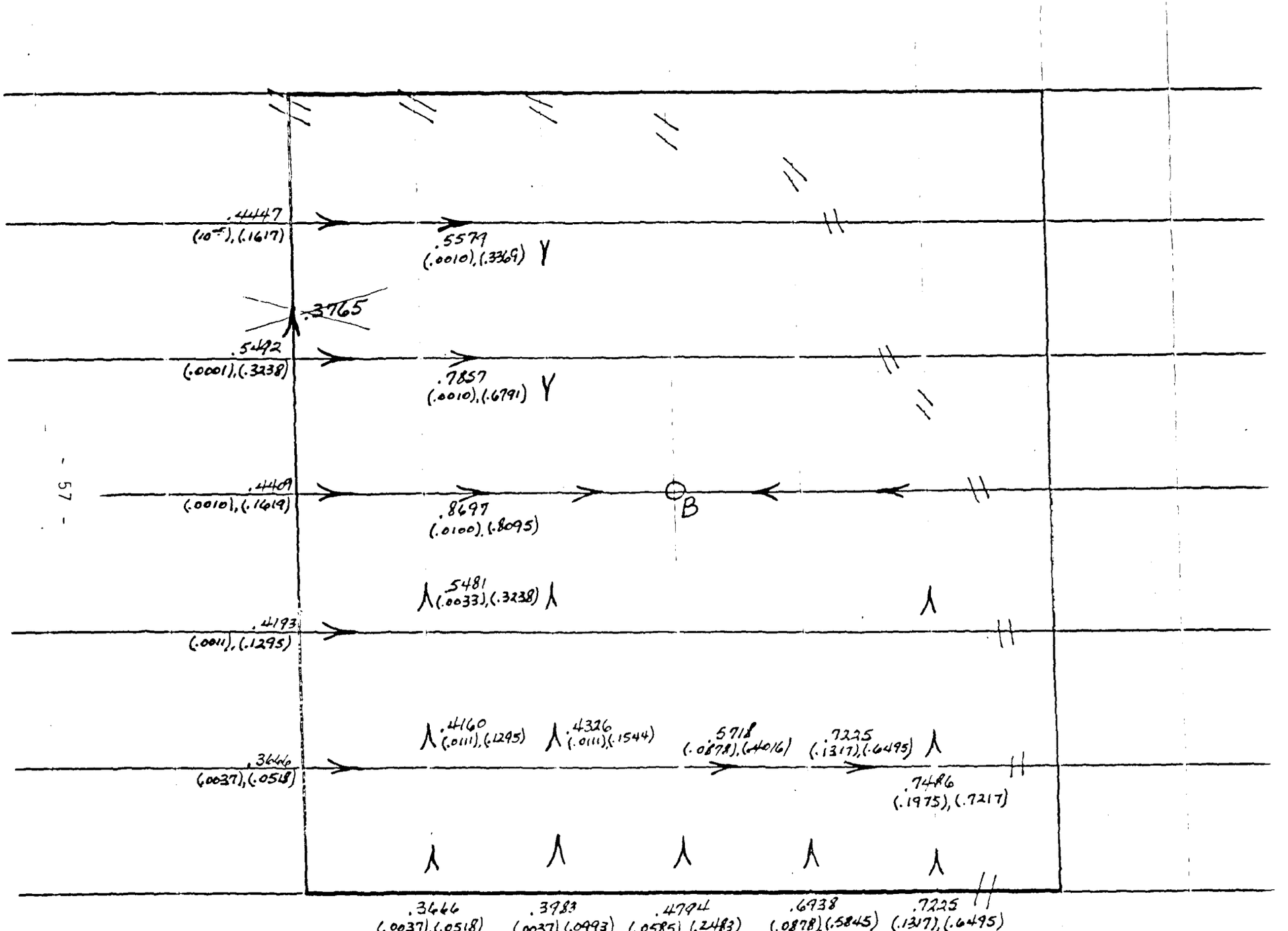

.4447
(10⁻⁵), (.1617)
.5579
(.0010), (.3369)
.3765
.5492
(.0001), (.3238)
.7857
(.0010), (.6791)
.4409
(.0010), (.1619)
.8697
(.0100), (.8095)
B
.5481
(.0033), (.3238)
.4193
(.0011), (.1295)
.4160
(.0111), (.1295)
.4326
(.0111), (.1544)
.5718
(.0878), (.4016)
.7225
(.1317), (.6495)
.3646
(.0037), (.0518)
.7486
(.1975), (.7217)
.3646
(.0037), (.0518)
.3983
(.0037), (.0493)
.4794
(.0585), (.2483)
.6938
(.0878), (.5845)
.7225
(.1317), (.6495)

FIGURE 24

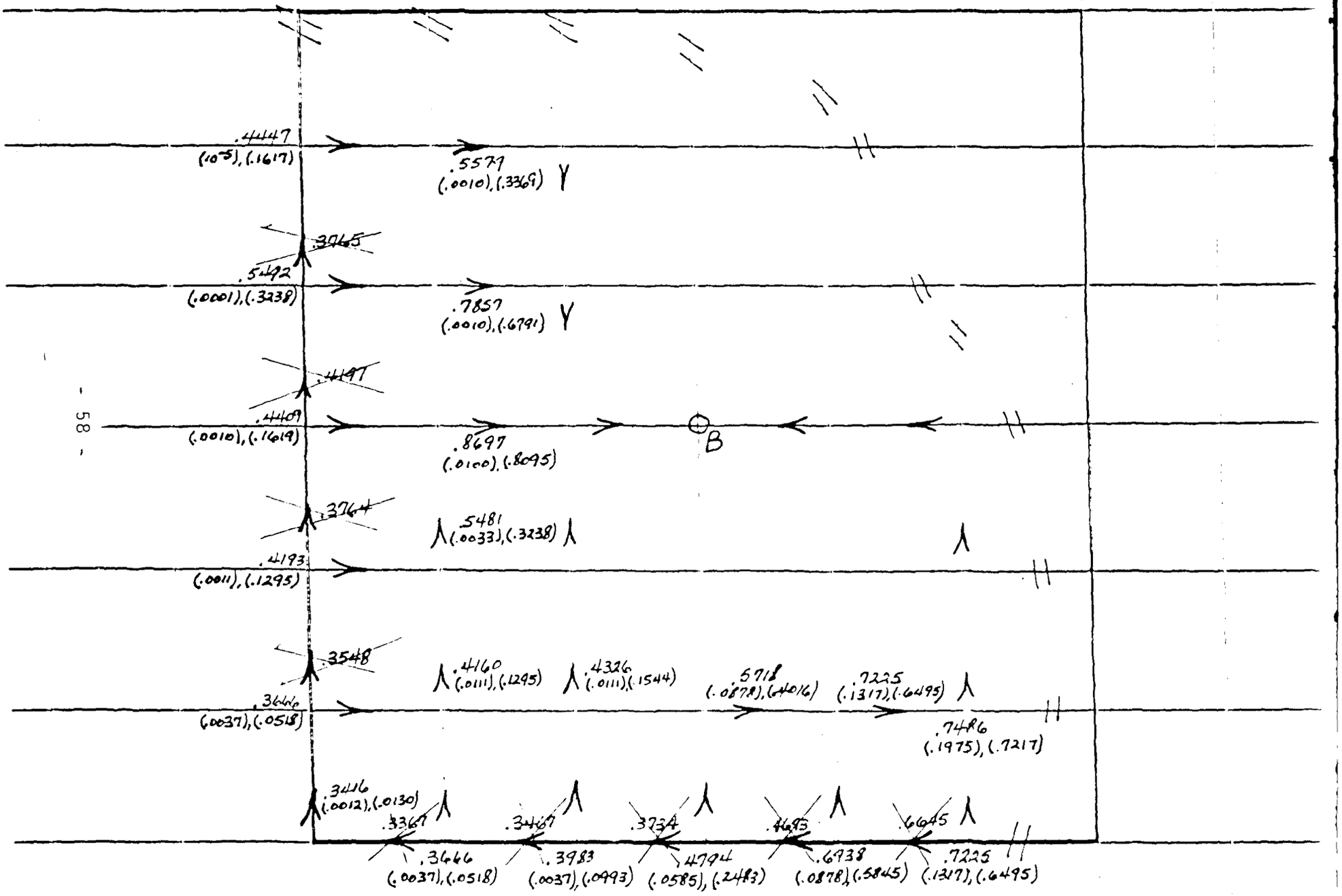

.4447
(.0005), (.1617)
.5579
(.0010), (.3369)
.3065
.5492
(.0001), (.3238)
.7857
(.0010), (.6791)
.4497
.4409
(.0010), (.1619)
.8697
(.0100), (.8095)
B
.3764
.5481
(.0033), (.3238)
.4193
(.0011), (.1295)
.3548
.4160
(.0111), (.1295)
.4326
(.0111), (.1544)
.5718
(.0878), (.4016)
.7225
(.1317), (.6495)
.3446
(.0037), (.0518)
.7486
(.1975), (.7217)
.3416
(.0012), (.0130)
.3367
.3467
.3734
.4693
.6645
.3466
(.0037), (.0518)
.3983
(.0037), (.0993)
.4794
(.0585), (.2483)
.6938
(.0878), (.5845)
.7225
(.1317), (.6495)

FIGURE 25

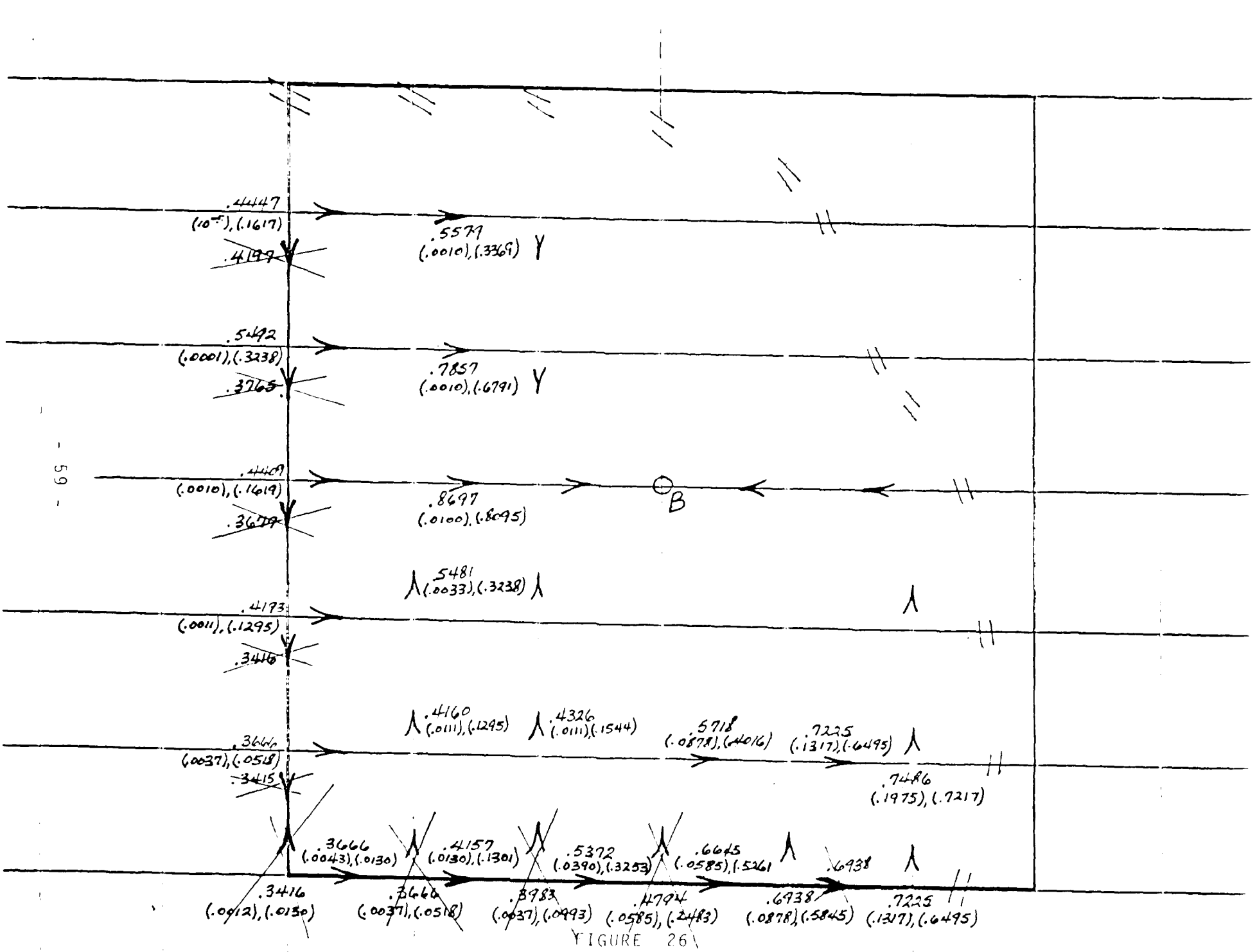

.4447
(10⁻⁵), (.1617)
.4197
.5579
(.0010), (.3369)
.5492
(.0001), (.3238)
.3765
.7857
(.0010), (.6791)
.4469
(.0010), (.1619)
.3679
B
.8697
(.0100), (.8695)
.5481
(.0033), (.3238)
.4193
(.0011), (.1295)
.3446
.4160
(.0111), (.1295)
.4326
(.0111), (.1544)
.5718
(.0878), (.4016)
.7225
(.1317), (.6495)
.3664
(.0037), (.0518)
.3415
.7486
(.1975), (.7217)
.3666
(.0043), (.0130)
.4157
(.0130), (.1301)
.5372
(.0390), (.3253)
.6645
(.0585), (.5261
.6938
.3416
(.0012), (.0130)
.3664
(.0037), (.0518)
.3983
(.0037), (.0993)
.4794
(.0585), (.2483)
.6938
(.0878), (.5845)
.7225
(.1317), (.6495)

FIGURE 26

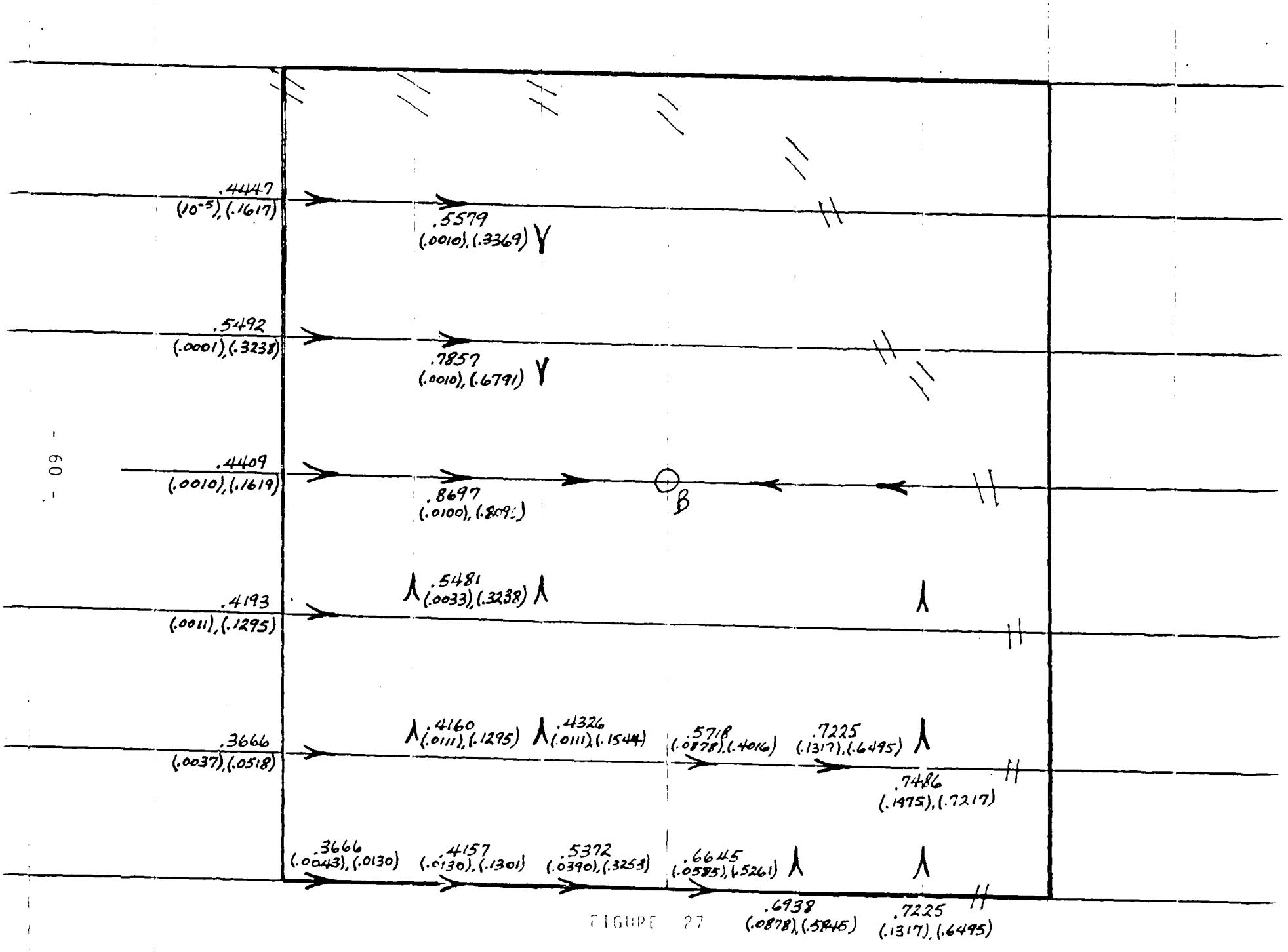

.4447
(10⁻⁵), (.1617)
.5579
(.0010), (.3369)
.5492
(.0001), (.3238)
.7857
(.0010), (.6791)
.4409
(.0010), (.1619)
.8697
(.0100), (.8095)
B
.5481
(.0033), (.3238)
.4193
(.0011), (.1295)
.4160
(.0111), (.1295)
.4326
(.0111), (.1544)
.5718
(.0878), (.4016)
.7225
(.1317), (.6495)
.3666
(.0037), (.0518)
.7486
(.1975), (.7217)
.3666
(.0043), (.0130)
.4157
(.0130), (.1301)
.5372
(.0390), (.3253)
.6645
(.0585), (.5261)
.6938
(.0878), (.5845)
.7225
(.1317), (.6495)

FIGURE 27

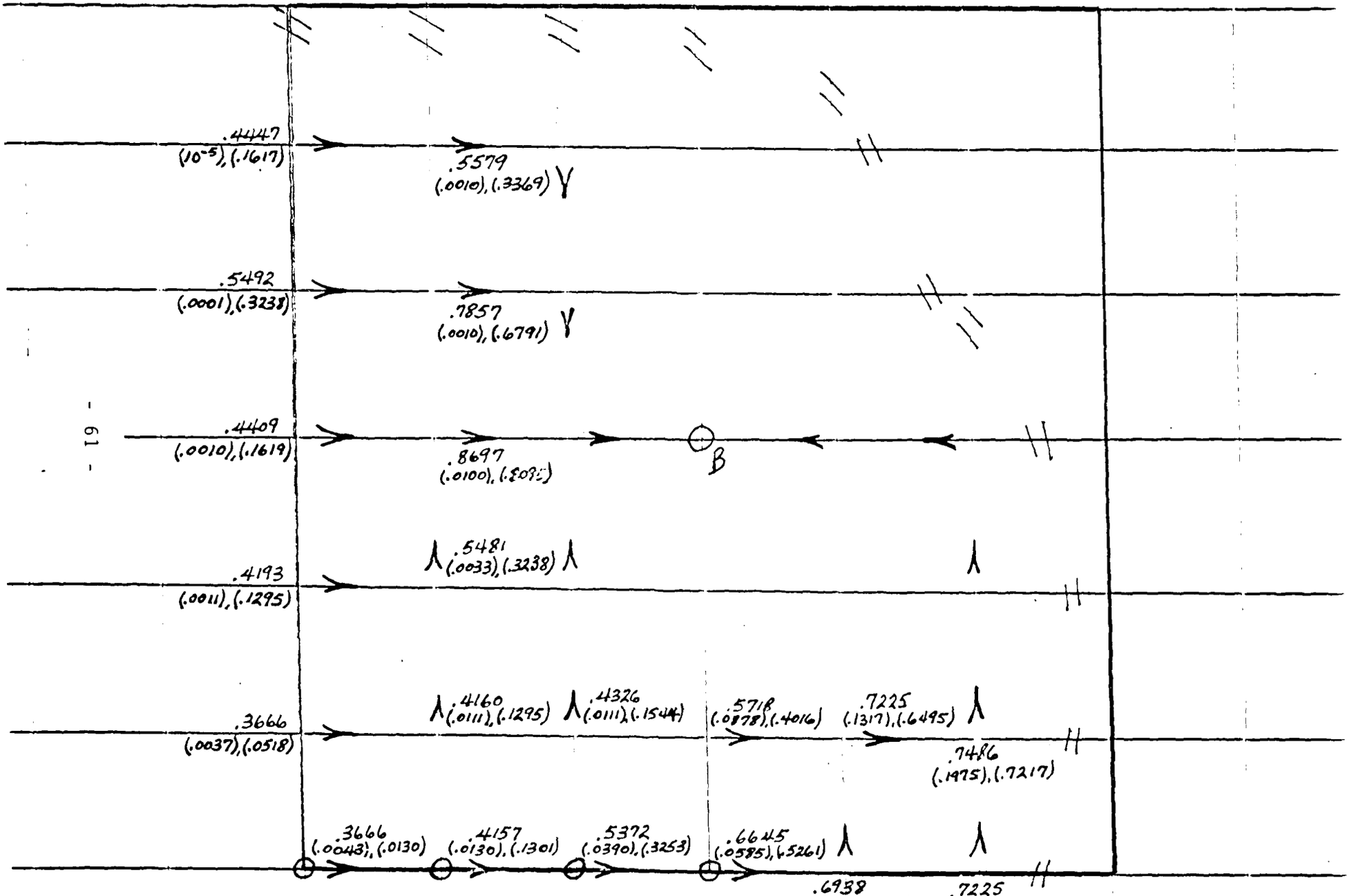

.4447
(10⁻⁵), (.1617)
.5579
(.0010), (.3369)
.5492
(.0001), (.3238)
.7857
(.0010), (.6791)
.4409
(.0010), (.1619)
.8697
(.0100), (.8095)
B
.5481
(.0033), (.3238)
.4193
(.0011), (.1295)
.4160
(.0111), (.1295)
.4326
(.0111), (.1544)
.5718
(.0878), (.4016)
.7225
(.1317), (.6495)
.3666
(.0037), (.0518)
.7486
(.1975), (.7217)
.3666
(.0048), (.0130)
.4157
(.0130), (.1301)
.5372
(.0390), (.3253)
.6645
(.0585), (.5261)
.6938
(.0878), (.5845)
.7225
(.1317), (.6495)

FIGURE 28

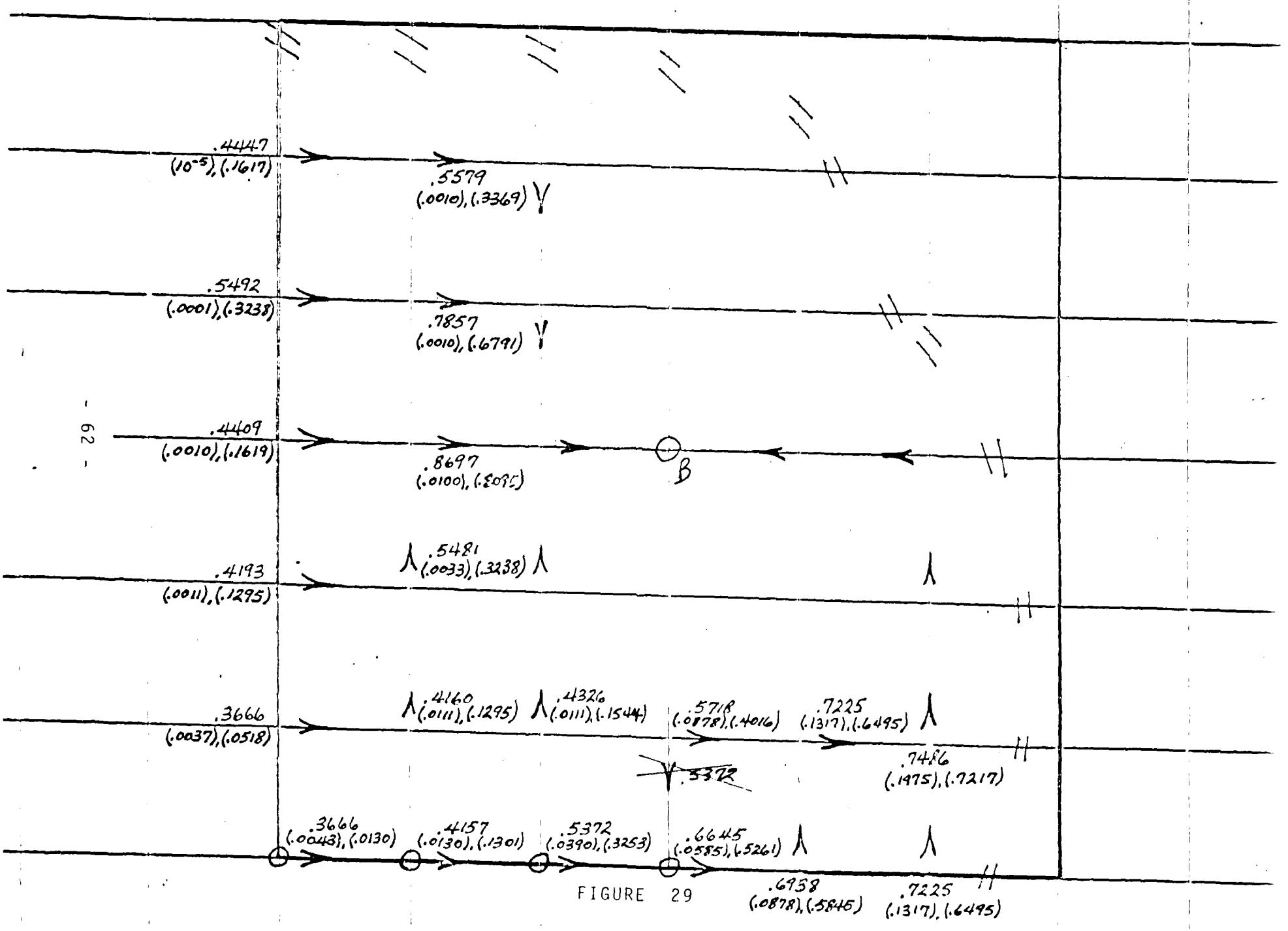

.4447
(10⁻⁵), (.1617)
.5579
(.0010), (.3369)
.5492
(.0001), (.3238)
.7857
(.0010), (.6791)
.4409
(.0010), (.1619)
.8697
(.0100), (.8095)
B
.5481
(.0033), (.3238)
.4193
(.0011), (.1295)
.4160
(.0111), (.1295)
.4326
(.0111), (.1544)
.5718
(.0878), (.4016)
.7225
(.1317), (.6495)
.3666
(.0037), (.0518)
.5372
.7486
(.1975), (.7217)
.3666
(.0043), (.0130)
.4157
(.0130), (.1301)
.5372
(.0390), (.3253)
.6645
(.0585), (.5261)
.6958
(.0878), (.5845)
.7225
(.1317), (.6495)

FIGURE 29

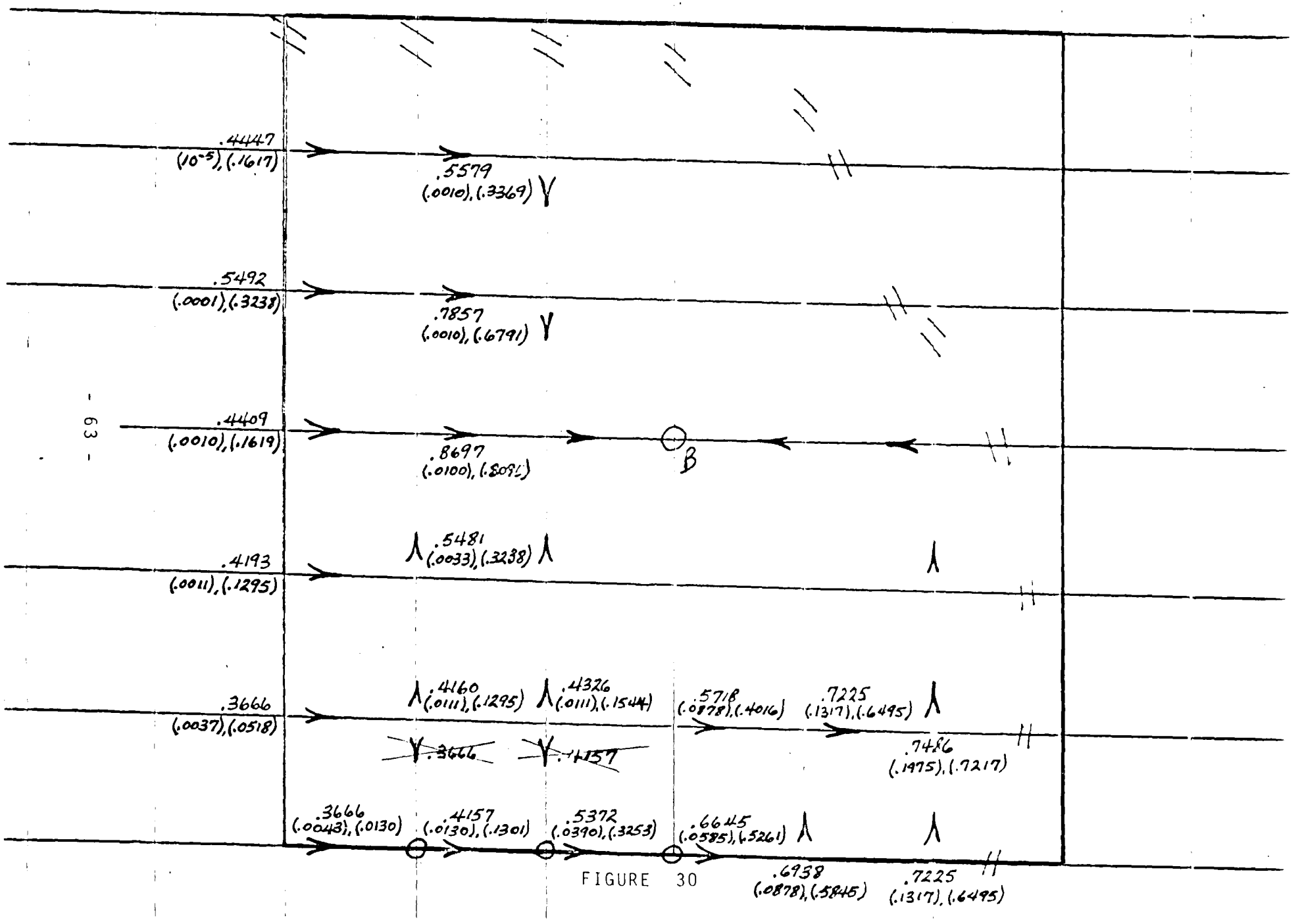
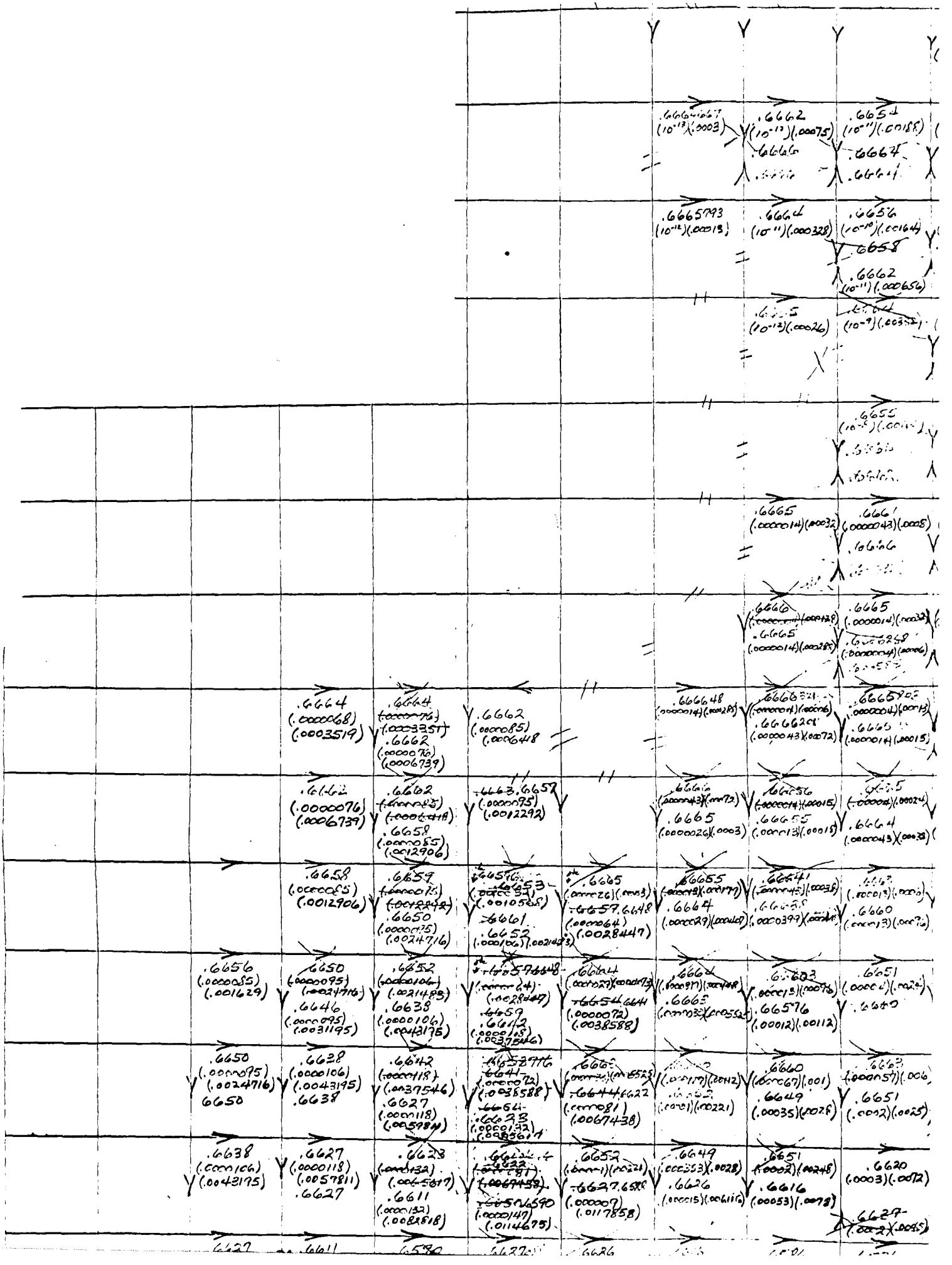
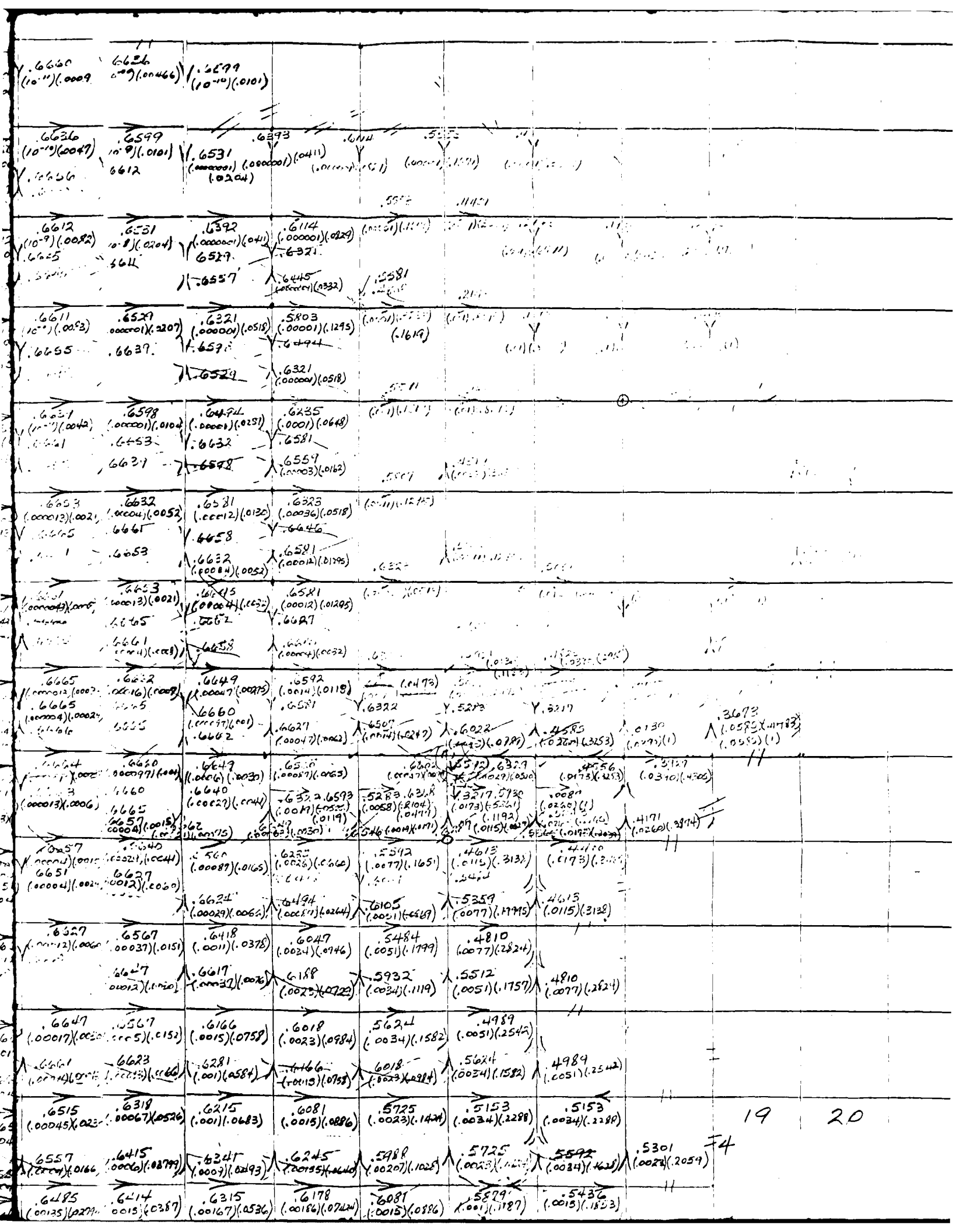
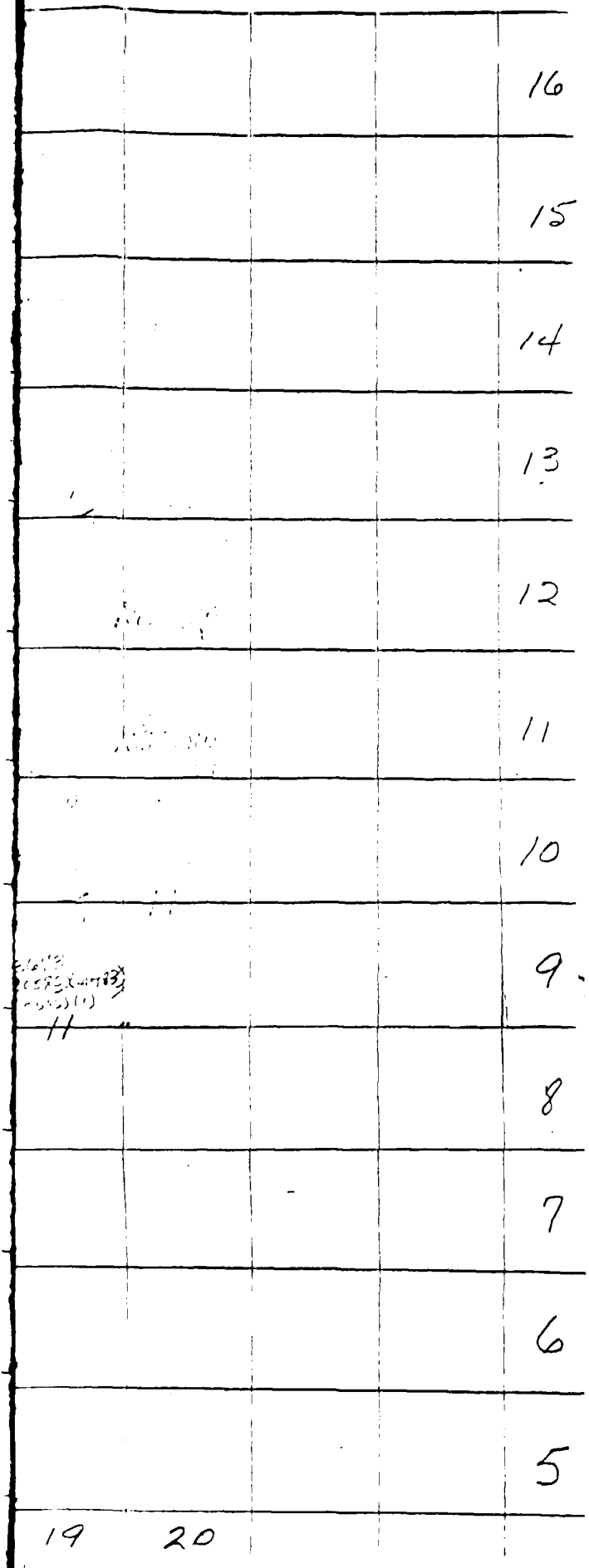
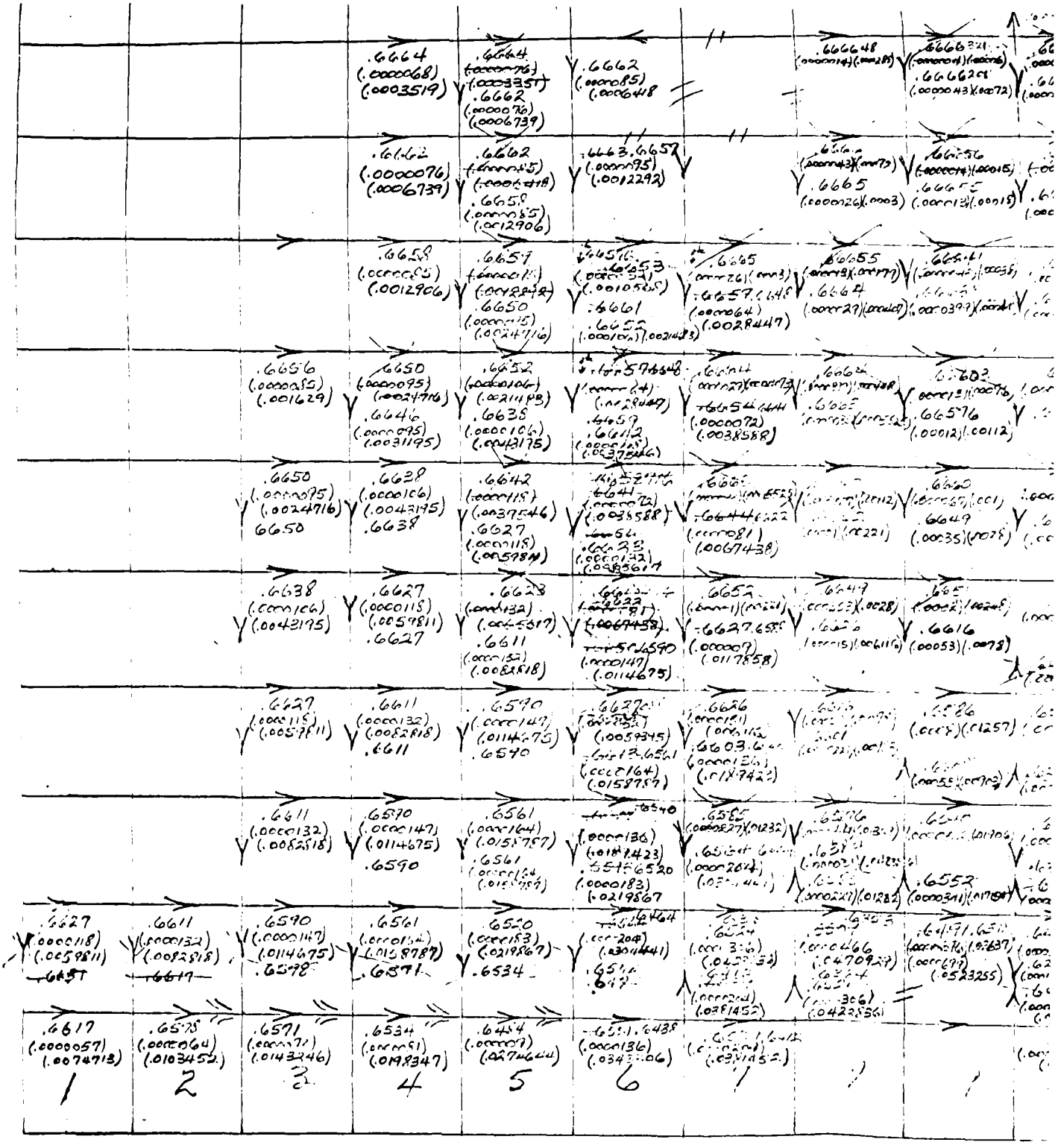

.4447
(10⁻⁵), (.1617)
.5579
(.0010), (.3369)
.5492
(.0001), (.3238)
.7857
(.0010), (.6791)
.4409
(.0010), (.1619)
.8697
(.0100), (.8091)
B
.5481
(.0033), (.3238)
.4193
(.0011), (.1295)
.4160
(.0111), (.1295)
.4326
(.0111), (.1544)
.5718
(.0878), (.4016)
.7225
(.1317), (.6495)
.3666
(.0037), (.0518)
.3666
.4157
.7486
(.1975), (.7217)
.3666
(.0043), (.0130)
.4157
(.0130), (.1301)
.5372
(.0390), (.3253)
.6645
(.0585), (.5261)
.6958
(.0878), (.5845)
.7225
(.1317), (.6495)

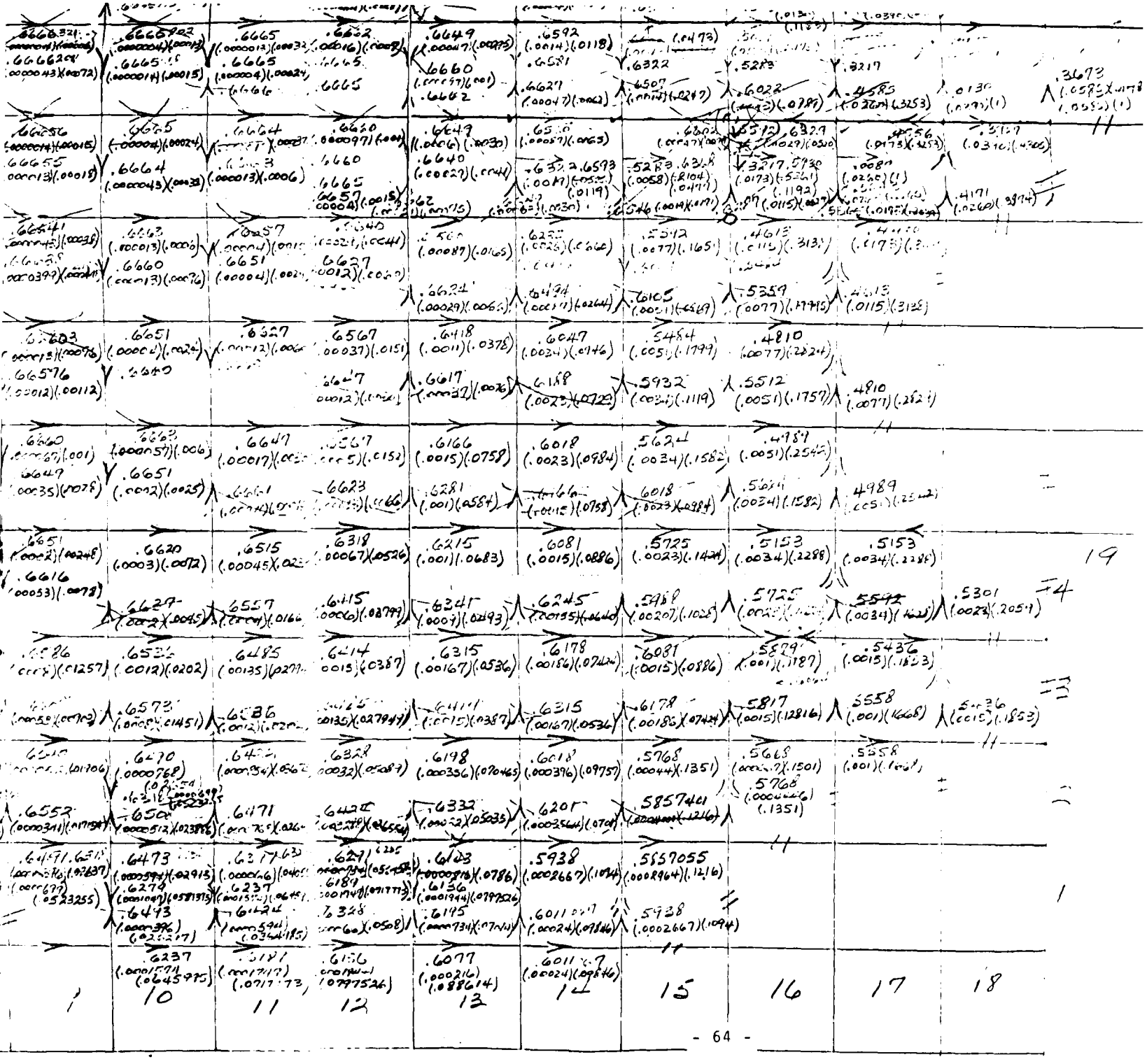
FIGURE 30




19
20


16
15
14
13
12
11
10
9
8
7
6
5
19
20





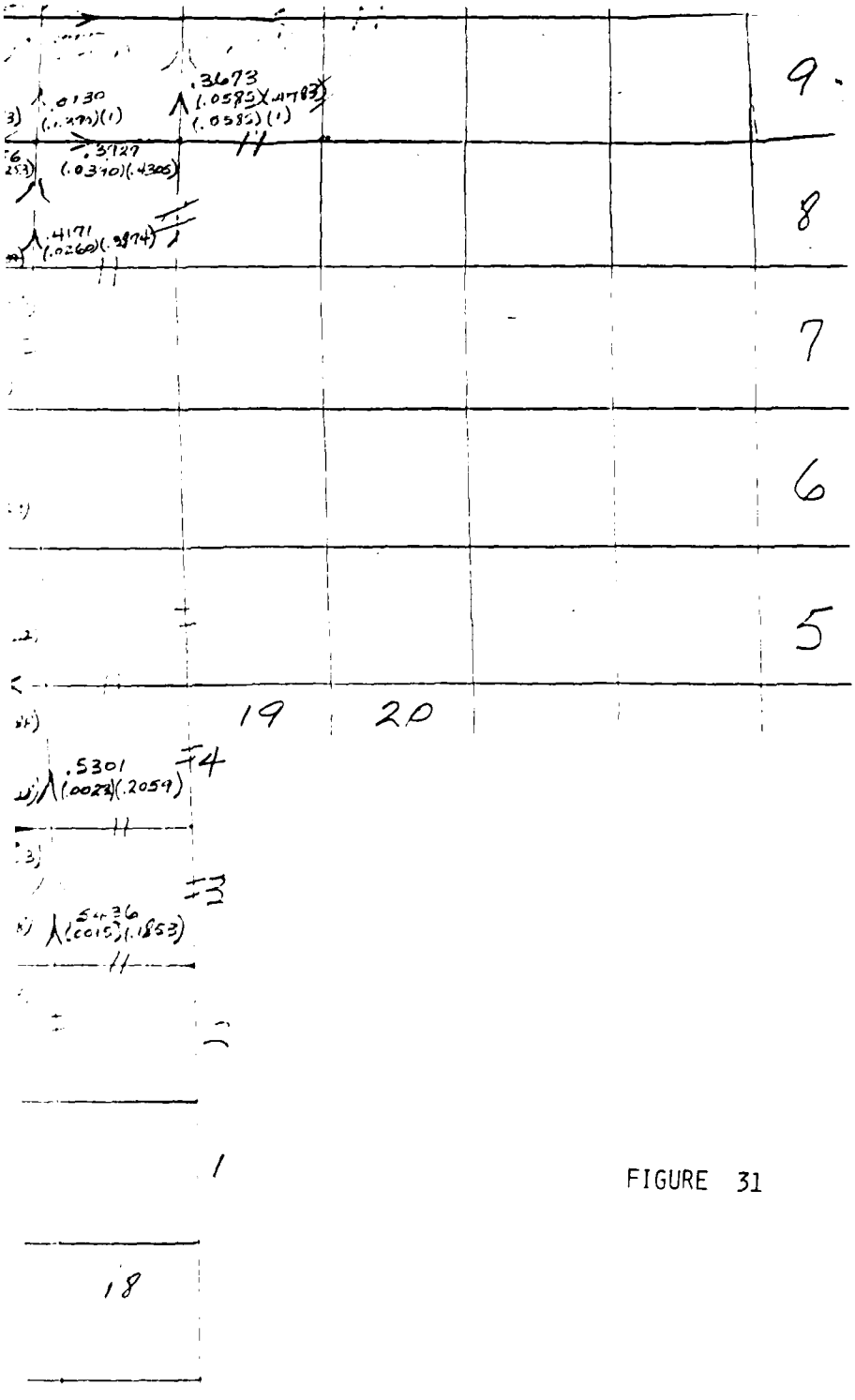
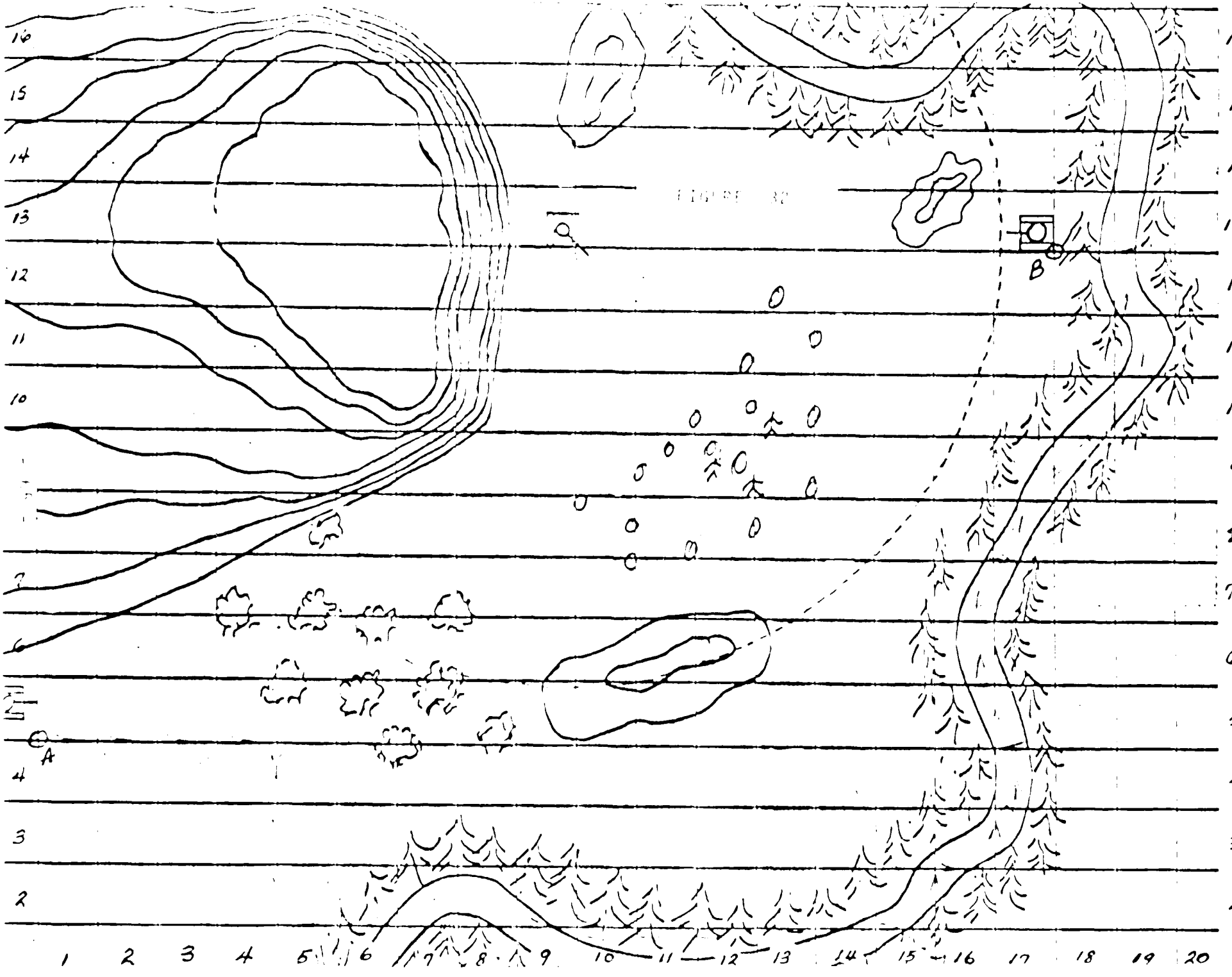

.0130
(.1479)(1)
.3673
(.0585)(.4783)
(.0585)(1)
.3129
(.0370)(.4305)
.4171
(.0260)(.9374)
9
8
7
6
5
19
20
.5301
(.0028)(.2059)
#4
#3
(.0010)(.1853)
1
18

FIGURE 31


A
B
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2

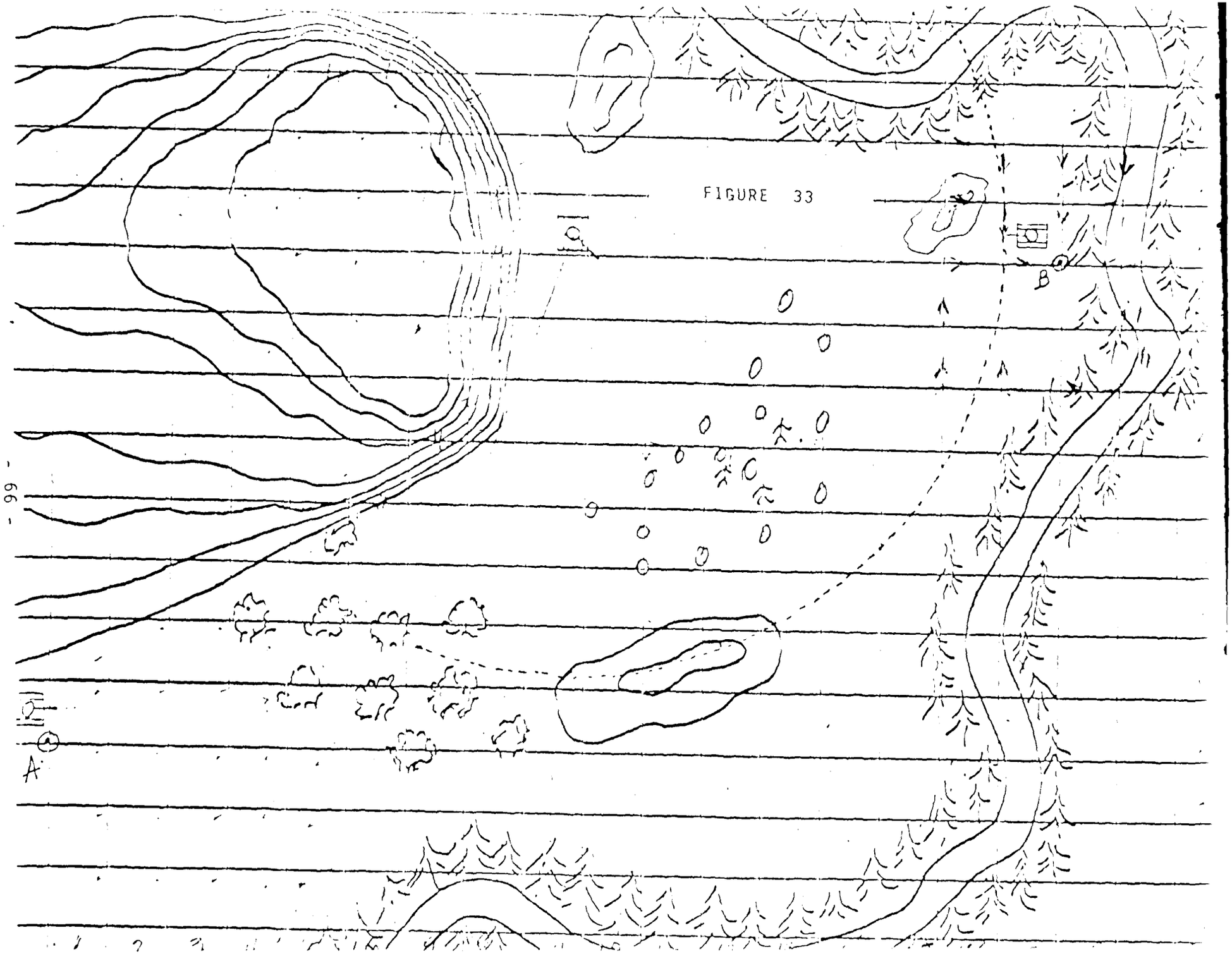


FIGURE 33

FIGURE 34

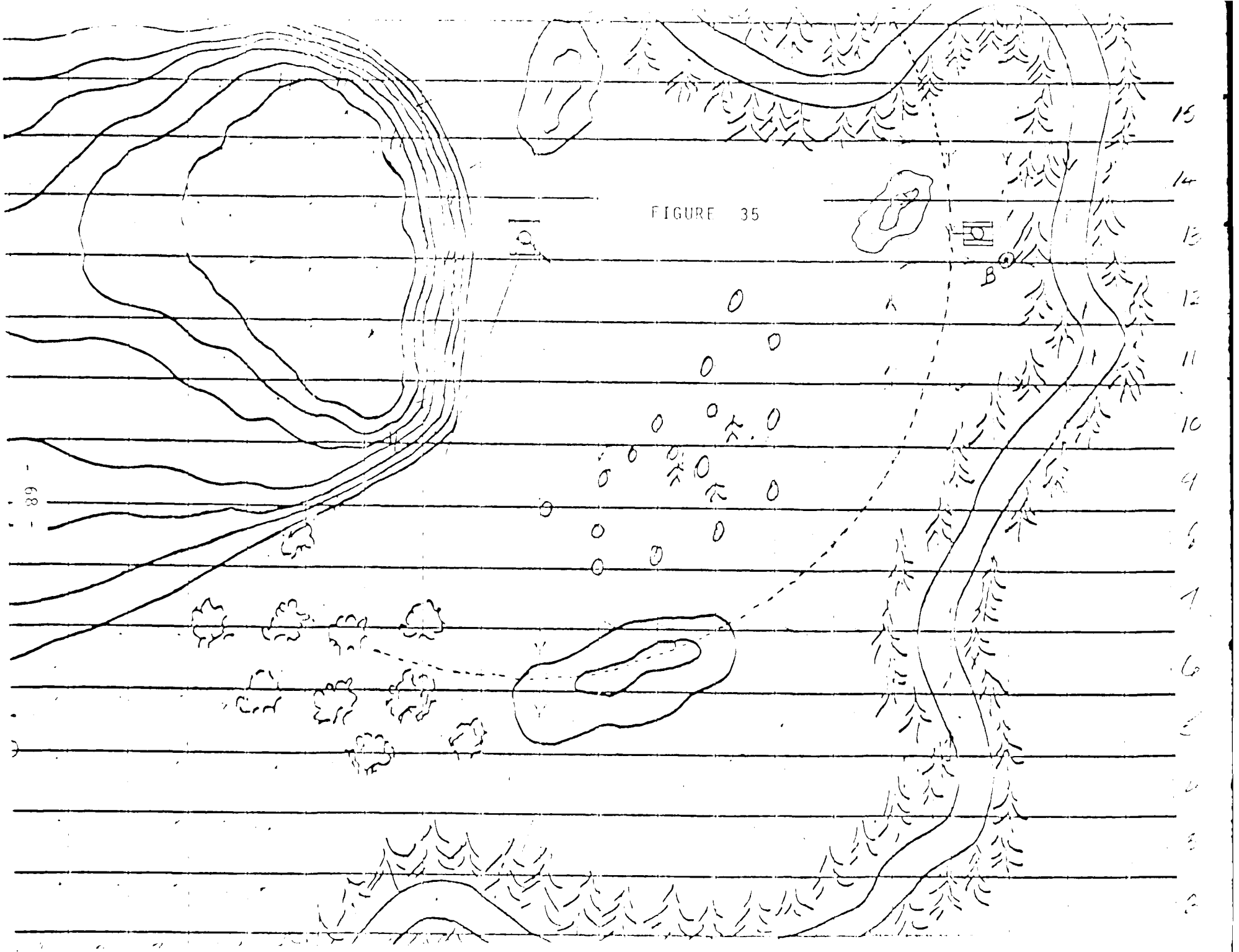


FIGURE 35

APPENDIX C

MEMORANDUM: AN ESTIMATION OF THE COMPUTATIONAL REQUIREMENTS FOR THE CALCULATION OF A MAXIMUM UTILITY PATH

MEMORANDUM

TO: A.J. OWENS DATE: April 13, 1982
FROM: G.H. BURGIN

SUBJECT: AN ESTIMATION OF THE COMPUTATIONAL REQUIREMENTS FOR THE CALCULATION OF A MAXIMUM UTILITY PATH

OBJECTIVE: The purpose of this memorandum is to estimate computer requirements for the calculation of a path along which a certain function (the path's utility function) is maximized. The estimate should provide both CPU time requirement as well as memory requirement for a tank battlefield of realistic size as it might be used in a simulator.

PROBLEM DEFINITION: We will assume a battlefield area of a square shape, divided into smaller squares where the smallest squares are sufficiently small to represent terrain features for the area. We will then number the vertices (or nodes) of the area starting at the left top corner (Northwest corner), proceeding West-to-East first, then North-to-South. Figure 1 illustrates a small sample battlefield to show the grid numbering scheme. In the following, the variable N will represent the number of vertices in the battlefield area, that is, the number of nodes in the network.

For the purpose of the maximum utility path algorithm, the AML tank will be allowed to move along branches in any direction. We will also assume that at a decision point (a point in time when the AML tank makes a decision about its future path), the tank is located at a vertex. At this point in time, the trainee's tank position is also given, and we will assume that knowing these two positions, we can calculate a utility associated with each branch. This utility is defined and described in the second progress report and can be expressed, for one individual branch, connecting vertex k with vertex ℓ as:

$$U_{k\ell} = K_1(1-DF_{T_{k\ell}}) + K_2 DF_{A_{k\ell}}$$

Similarly, we may define the utility of the branch connecting vertex ℓ with vertex m to be

$$U_{\ell m} = K_1(1-DF_{T_{\ell m}}) + K_2 DF_{A_{\ell m}}$$

The total path utility for going from vertex k to vertex m through the node ℓ is defined to be

$$U_{k\to m} = K_1(1-DF_{T_{k\ell}} \cdot DF_{T_{\ell m}}) + K_2 DF_{A_{k\ell}} \cdot DF_{A_{\ell m}}$$

For a path extending over n branches, the utility therefore is:

$$U = K_1\left(1-\prod_{i=1}^{n} DF_{T_i}\right) + K_2 \prod_{i=1}^{n} DF_{A_i} \qquad (1)$$

The problem, therefore, can be expressed succinctly as:

Given a starting node P_S and a destination node P_D, find the path connecting P_S with P_D for which (1) will be maximized.

SCOPE: The problem of finding the path with maximum utility is really composed of two problems. One is the determination of the "survivability" values and the "attackability" values along each branch in the network given the positions of the two tanks. There is obviously some computational effort associated with this part of the problem. This memorandum does not address this problem. We are only concerned with the computational requirement for the second part of the problem which is: Given utility values for each branch, find the optimal path from a given source to a given destination.

APPROACH: The optimal path algorithm as developed in the second progress report, which we will call the "expanding square" algorithm, may be computationally efficient; however, it is very difficult to estimate the time required to find a path. This is due to the fact that when proceeding to an expanded square, the optimal solution found so far may no longer be optimal, and the calculations performed so far for the inner squares may have to be, at least partially, repeated. We see no way to estimate how often this situation would occur. We consider an algorithm, therefore, which may (or may not) be computationally less efficient but which permits an estimate of an upper bound of computational effort. This algorithm is

an adaptation of a "shortest route" algorithm as described in Operations Research: A Managerial Emphasis by Ronald V. Hartley, Goodyear Publishing Company, Pacific Palisades, California, 1976.

DESCRIPTION OF THE BASIC ALGORITHM: We will first describe the algorithm in its original form in which the problem is to minimize the length of the path where the individual lengths of the branches are simply added. It is important to realize that the shortest route algorithm was developed for a network of arbitrary topology. The fact that our network for the tank problem has a very special structure can be exploited and will provide a great savings in memory requirements. As in dynamic programming (the algorithm is not a dynamic programming technique, however), in addition to the solution of the original problem (find maximum utility path between node P_S and P_D), the algorithm actually provides a solution to the more general problem: "Find the maximum utility path between the starting node and any arbitrary other node".

The algorithm is best explained by an example. Consider the network shown in Figure 2, and let us assume that we want to find the shortest path from node 1 to node 16.

PRELIMINARY STEP: - Number the nodes as outlined above and write down the matrix M showing distance, d_{ij}, between adjacent nodes i and j. (FIGURE 3).

STEP 1: - Define i equal to the starting column ($i=1$).

STEP 2: - Set $U_i = 0$; cross out column i. (In our example, $i=1$, so $U_1 = 0$ column 1 crossed out). Whenever a U_i (shortest path) is determined for a node, that value is placed into column $(N+1)$ of row i as shown in FIGURE 4.

STEP 3: - Define a set S.

$S = \{(U_i + d_{ij})$ such that U_i exists and j is not yet crossed out.$\}$

STEP 4: - Set MIN equal to the smallest element from S and j^* equal to the j of the smallest (U_i+d_{ij}) from the set S and i^* to the i of the smallest (U_i+d_{ij}) from the set S. If several (U_i+d_{ij}) have the same value, choose the one with the smallest value of i, next the smallest value of j.

STEP 5: - Set $U_i = U_{j^*} = MIN$ where $i=j^*$. Place U_i into column $(N+1)$ of row i and i^* into column $(N+2)$ of row i.

STEP 6: - Cross out column j^*.

STEP 7: - If not all U_i are determined, return to STEP 3, else proceed to STEP 8.

STEP 8: - The problem is now practically solved; the remaining task is to backtrack the solution from any desired destination node to the source node. This can be done as follows:

a) Set i to the desired destination node number.
b) Look up i^* from column $(N+2)$ in row i.
c) Set $i=i^*$ and go back to b) until i=source node number.

An example of this algorithm is carried out in detail in Appendix A of this memorandum. Applying STEP 8 yields:

i	i*
16	15
15	14
14	10
10	11
11	7
7	6
6	2
2	1

The optimal path from 1 to 16 therefore is:

1-2-6-7-11-10-14-15-16

An intuitive "proof" of the algorithm is presented in Appendix B of this memorandum.

MODIFICATIONS TO THE BASIC ALGORITHM: The first modification is the addition of two more rows and two more columns to the original network. This will make all the vertices of concern interior vertices, and in estimating computational requirements, the end-effects will not enter into the problem; all nodes will have exactly four branches emanating from them. Therefore, each row of the original matrix will have exactly four d_{ij}'s defined.

The second modification concerns the calculation of the U_j^* values and the optimization criterion.

$$U_{j*} = K_1 \left(1 - \prod_{i=1}^{n} DF_{T_i}\right) + K_2 \prod_{i=1}^{n} DF_{A_i}$$

Stored with each U_{j*} value will also be:

$$P1_{j*} = \prod_{i=1}^{n} DF_{T_i}$$

and

$$P_{2_{j*}} = \prod_{i=1}^{n} DF_{A_i}$$

The set S in the algorithm now consists of

$$S = \left\{(D_{i,j})\right\}$$

where $D_{i,j}$ is calculated as follows:

$$Q_1 = P_{1_i} * DF_{T_{i,j}}$$

$$Q_2 = P_{2_i} * DF_{A_{i,j}}$$

$$D_{i,j} = K_1 (1-Q_1) + K_2 Q_2$$

The calculation of $D_{i,j}$ therefore requires:

4 multiplications
1 addition
1 subtraction

MEMORY REQUIREMENTS: - Consider first the requirements for storing the matrix of the survivability and attackability values for each branch of the network. Basically, this matrix consists of N rows and N columns (remember, N is defined as number of nodes in network); but due to its special form, it is not necessary to provide N^2 cells for this matrix. Basically, each row contains four survivability values and four attackability values for the four branches from the node to its surrounding nodes.

In addition, the algorithm requires the storage of the products P_{1_i} and P_{2_i} and of U_i and i^* for each row.

Thus, the total memory requirement for data storage for the maximum utility path algorithm will be

$$S_{total} = N\,[(4+2) + 4] + \text{temporary storage} = 12\,N + \text{temporary storage}$$

For a network consisting of 256 nodes, this amounts to only about 3,000 words, which is certainly less than will be required to store the data describing the terrain features.

Memory requirement for the program of the maximum utility path algorithm is estimated not to exceed 3,000 words of 16 bit length; total random access memory requirement, including temporary data storage is estimated not to exceed 8k.

CENTRAL PROCESSOR TIME REQUIREMENTS: - We will look at this problem in the following way: Given a realistic tank simulation battlefield grid and given typical execution times of

today's commercially available microprocessors, how much CPU time would be required to perform a path optimization?

Figure 5a shows a typical battlefield grid, wherein the vertices formed by the heavy lines are used for path optimization while the terrain data is given both at the vertices formed by the heavy lines and the light lines. Once the two tanks have approached each other so that they are closer than some threshold distance, the algorithm would switch and use the finer grid for path optimization.* The network to be used in our time estimate will consist of 256 nodes.

Figures 6 and 7 further illustrate the fact that a grid with 16 divisions at 300m length each for path selection subdivided into subsquares of 150m x 150m is sufficient. Figure 6 shows a typical, populated area in Europe, while Figure 7 is representative for a desert area.

To obtain a time estimate, we need first an estimate of the average number of the members of set S which we will call $\bar{N}_S$. This number will be four at the first iteration and then grow as the algorithm proceeds. When about half of the rows have been processed (that is, an U_i value has been determined for about 128 nodes), the value of N_S will be a maximum and from thereon will decrease until it reaches again 4 at the end of the algorithm.

If we have 128 U_i's defined, the maximum (worst case) number of elements in the set S would be

$$N_{S_{max}} = (123*3) + 1 = 370$$

*As shown in Figure 5b.

Taking N_S as the average between $N_{S_{max}}$ and $N_{S_{min}}$ yields

$$N_S = \frac{370 + 4}{2} = 187$$

Thus, on the average, for each one of the 256 rows of matrix M we would have to perform 187 D_{ij} evaluations, amounting to:

187*4 = 748 multiplications

187 additions

187 subtractions

Finding the largest D_{ij} value requires an additional:

187 comparisons

93 replacements

Using representative execution times

t_m = 6 μs (multiplication)

$t_a = t_s$ = 3 μs (addition and subtraction)

$t_c = t_r$ = 2 μs (comparison and replacement)

results in

$$T_i = 748*6 + 187\ (3+3+2+1) = 6,171\ \mu s$$

For one optimization, we have to complete 256 individual iterations, so that

$$T_{total} = \sum_{1}^{256} T_i = 1,579,776\ \mu s = 1.5\ sec$$

The above number of operations did not include fetching and storing of operands (which requires extensive subscript calculations, but can all be performed in integer arithmetic). To obtain a conservative time estimate we will multiply the above time by a factor of five, which then results in a total CPU time requirement of 7.5 seconds for one complete path-optimization in a 256 node network.

This time estimate is conservative for three reasons:

1. The maximum number of members in the set S may never be 370.
2. Some of the address calculations can be performed by operations whose execution time is less than one microsecond.
3. The algorithm, in general, can be terminated as soon as U_i for the destination node is found. That means that not all of the N rows of matrix M have to be processed; thus, in general, less than N iterations are required.

SOFTWARE AND HARDWARE COST CONSIDERATIONS: The preceding sections demonstrated the technical feasibility of realtime path optimization using presently commercially available microprocessors. The iAPX 432 was selected as an example because it offers the greatest convenience in programming; it is fast enough so that floating point arithmetic may be used whenever it is convenient to do so. Certainly, in developing an AML-driven intelligently interactive tank, it

is desirable to develop the algorithms in a high level language, using floating point variables of at least 32 bits length. However, when the AML algorithm is used in a production model (that is, the algorithm is essentially completely developed, and all that may change are different data bases representing various battlefield terrains), most of the path optimization algorithm could be executed using integer arithmetic. By carefully scaling the variables, most, or maybe all, the calculations can be performed with 16 bit integers. As a consequence, computer hardware costs will be decreased by about an order of magnitude. For example, an Intel 43201 chip (which is the most expensive of the three chips, making up the iAPX 432), costs about $800 today while a comparable 16 bit microprocessor, the TMS 99110 by Texas Instruments, sells for about $100 now. (See Appendix D).

These two prices, $800 and $100, are prices for a single chip (in quantities of 100); a complete computer having enough capabilities to perform the tasks required to drive the AML tank requires a number of additional chips, such as input/output interface chips, disk controller, memory mapper plus the necessary random access memory chips.

Discussions with Intel salespeople provided the following estimates of the cost of a complete Intel iAPX based system (occupying about the volume of a shoe box): projected for 1984--about $25,000. If such a system is bought in quantities of 100, the cost would be reduced to about $15,000.

By the year 1984, Intel will have available a new microprocessor, the iAPX 286, which has a 16 bit word size. It corresponds in capability approximately to a Digital Equipment Corporation PDP 11/70. The basic chip cost for the iAPX 286 is $250. A complete computer system is estimated to cost about $2,500. This seems to be a realisitc estimate of the price of the computer required in a production unit of the AML-tank simulator.

PROGRAMMING LANGUAGE: The development version of the AML-tank driver may be programmed in any suitable high-level language. Preferred languages would be Pascal or Fortran; Pascal because it is closer to Ada than Fortran, while the advantage of Fortran would be its presently widespread use. The production version should be programmed in Ada because this is the language preferred by DoD. Intel Corporation has presently contracted to an outside company the development of an Ada compiler for the iAPX 286, which, I was told, will be ready in 1984. An Ada cross compiler for the iAPX is already available with the VAX as a host machine.

It is proposed that the development of a production version of this algorithm be performed in two or three steps, depending on the availability of the iAPX 286 and its Ada compiler.

1. Implement the algorithm on a commercially available time-shared system (preferably a VAX which has a basic word size of 16 bits but supports 32 bit real variable operations) in Fortran or Pascal.
2. If after successful completion of a development version in step 1 the iAPX 286 and its Ada compiler are avaiable, proceed to convert the AML-tank simulator program to Ada, using cross compiler on the VAX. If they are not available, convert to Ada, using an iAPX 432 cross compiler.
3. Finalize the iAPX 286 version with the intent to minimize computer hardware cost, such that the computer cost would be in the order of $2,000 to $3,000.

SUMMARY AND CONCLUSIONS: Applying a modification of the classical shortest route algorithm, it is possible to estimate the number of arithmetic operations required to find the path with maximum utility from any specified source vertex to any given destination vertex.

An iAPX 432 based system can perform path-optimization for a realistic battlefield in approximately five seconds using floating point arithmetic.

An iAPX 286 based system can perform the same task in about the same time using all integer arithmetic.

Memory requirements for the algorithm do not exceed 8k of 16 bit words.

A self-contained, iAPX 432 based development system would cost about 25,000 dollars.

The computer cost for an iAPX 286 based production unit, projected to 1984, would be of the order of $2,000 to $3,000.

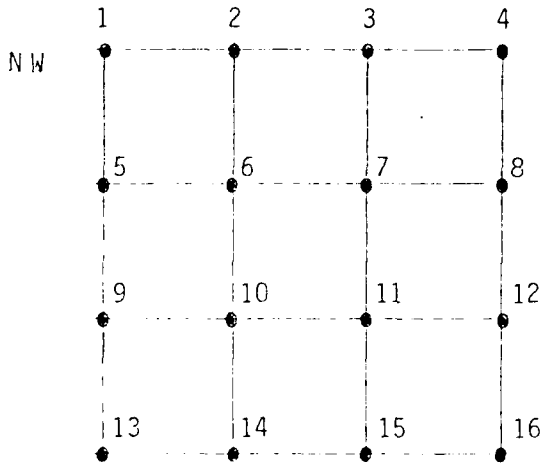


FIGURE 1: THE BATTLEFIELD GRID

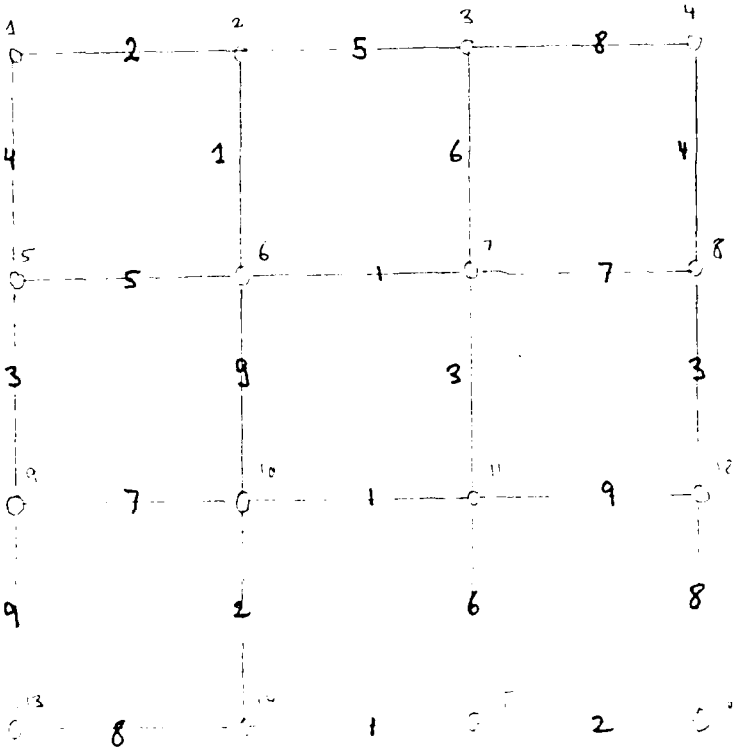


FIGURE 2: A SAMPLE "BATTLEFIELD"

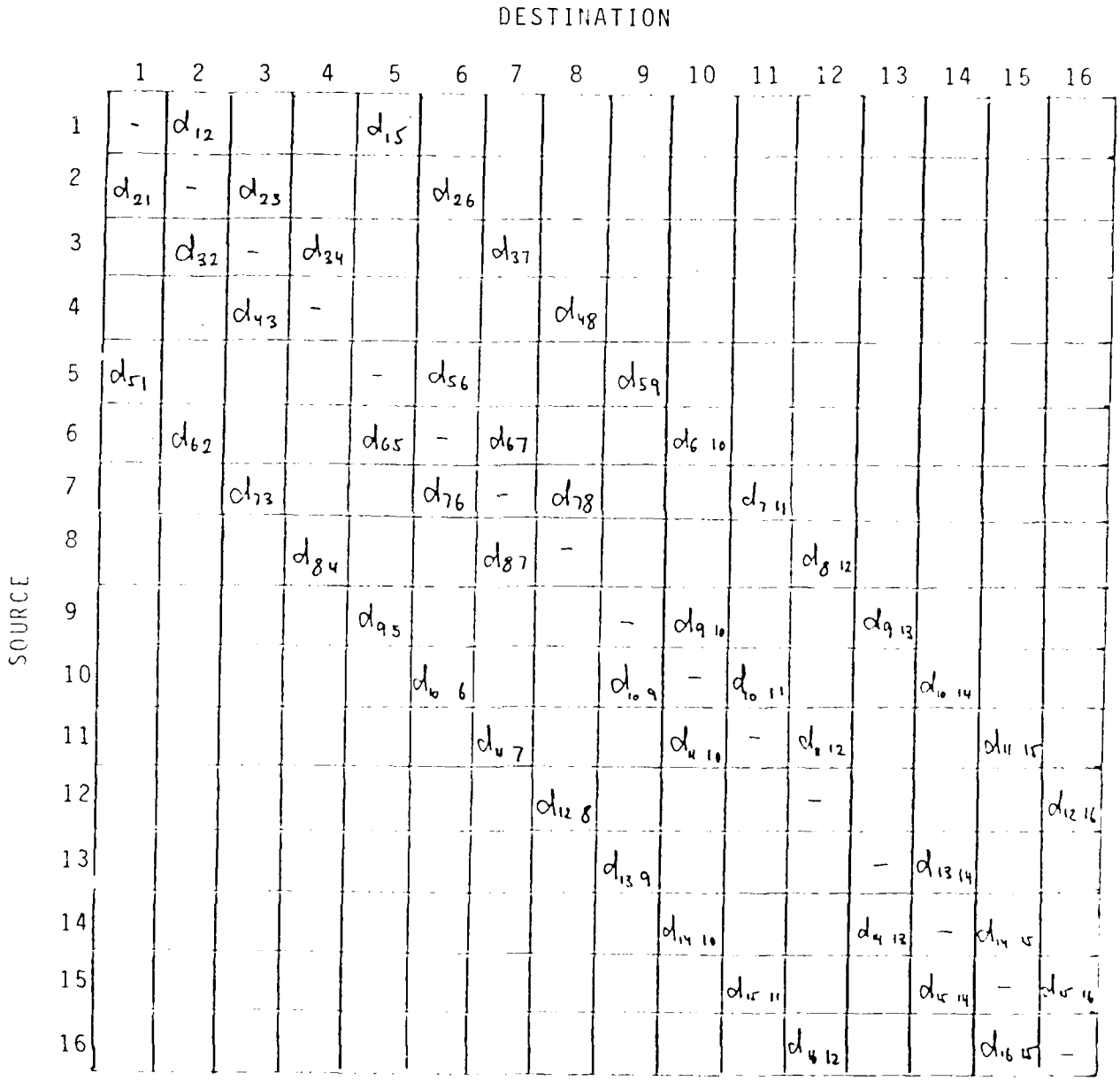


FIGURE 3: THE GENERAL SHAPE OF THE M-MATRIX.

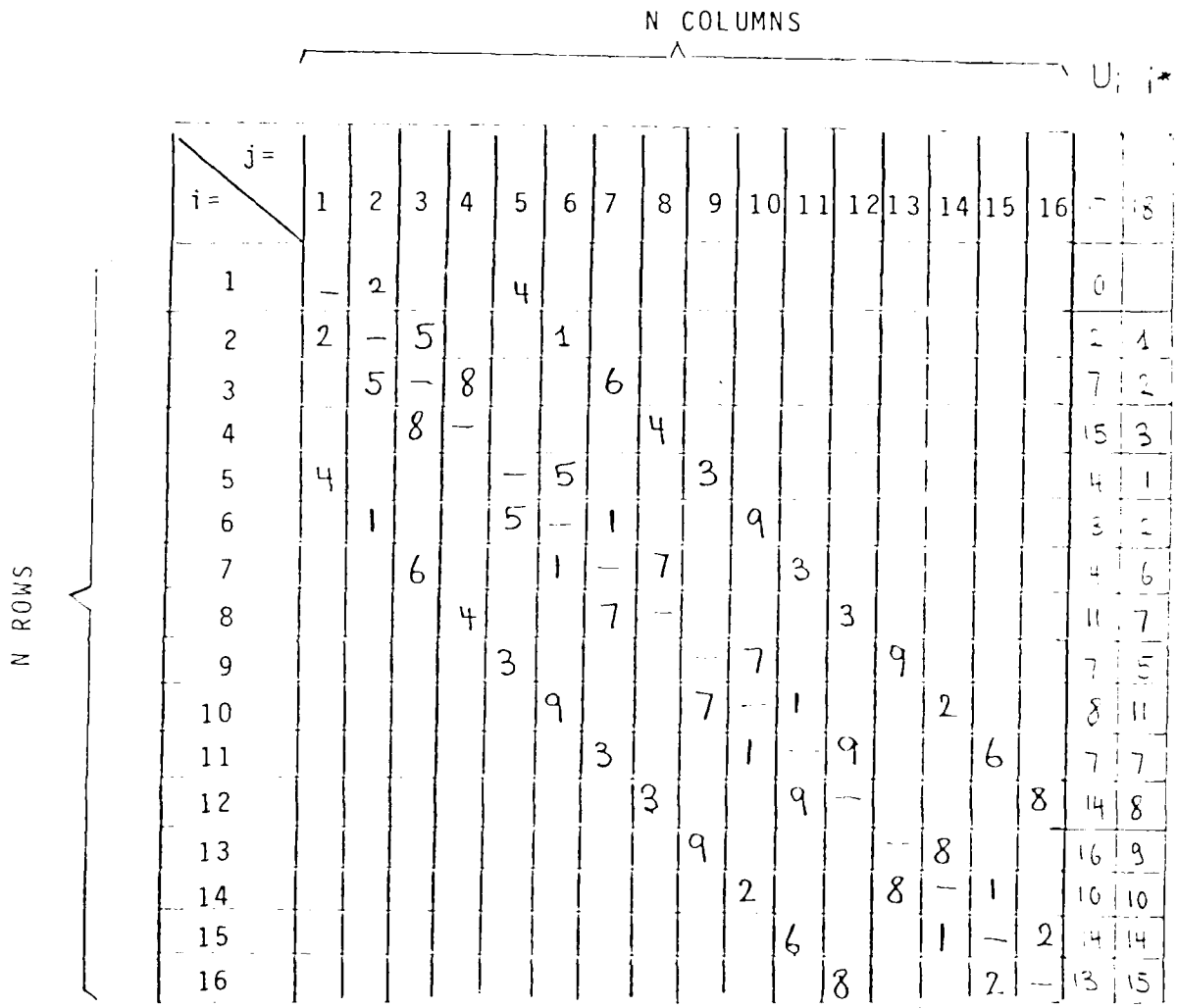


FIGURE 4: The (symmetrical) M matrix for the sample problem.

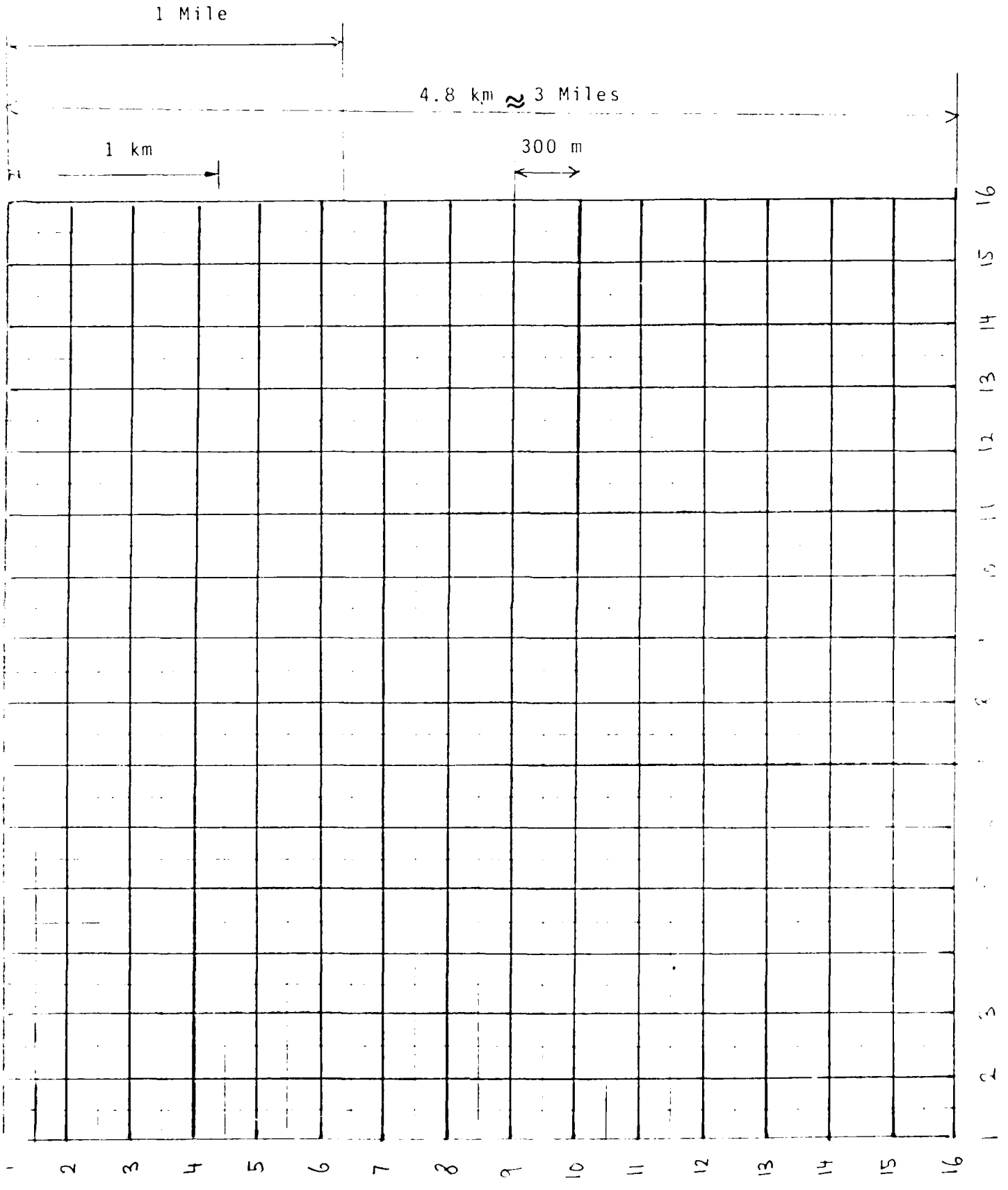


Figure 5a:- Proposed Subdivision of Battlefield Area.

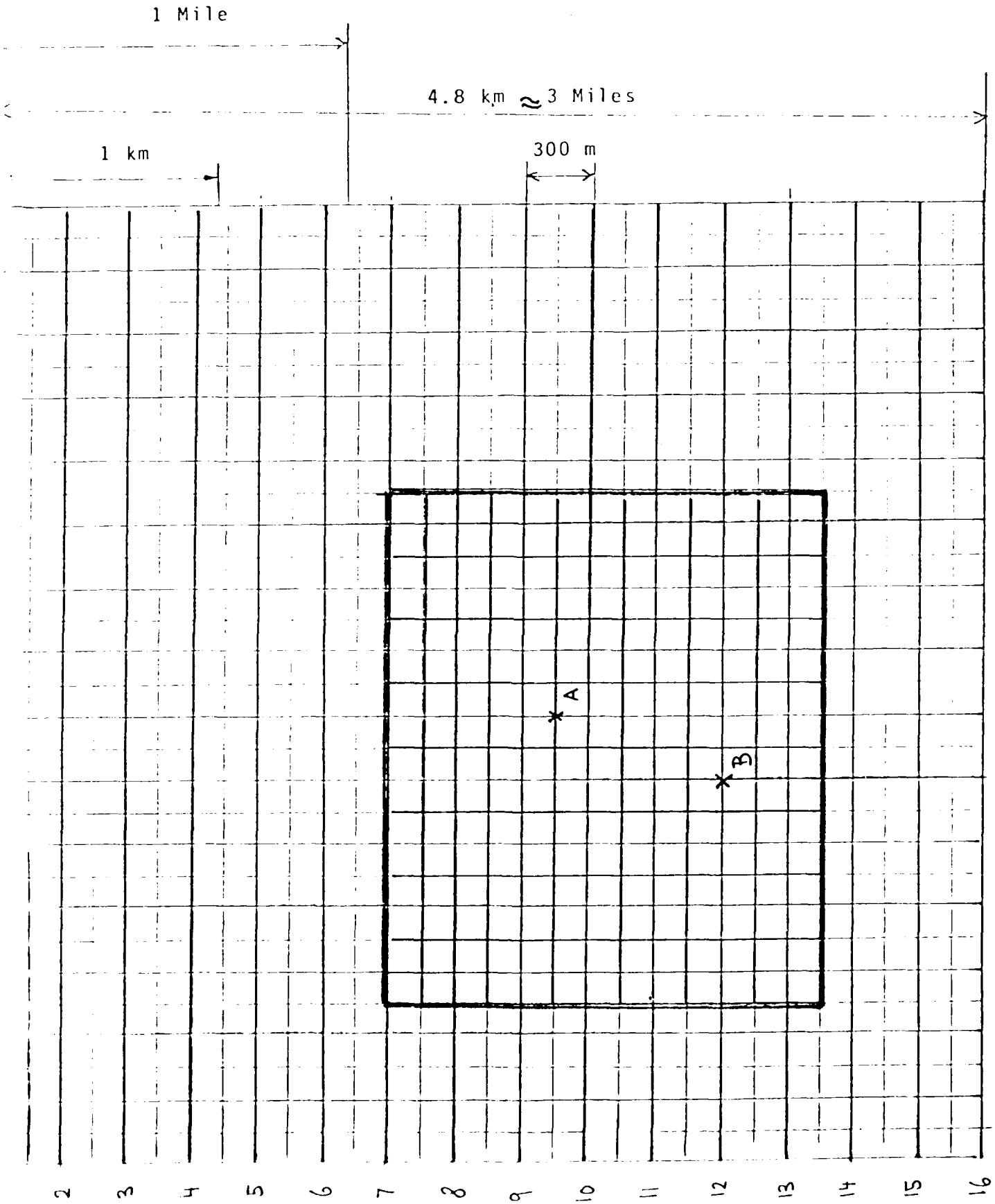


Figure 5b: - The Reduced Size Battlefield Area When the Two Tanks are Within a Certain Distance of Each Other.

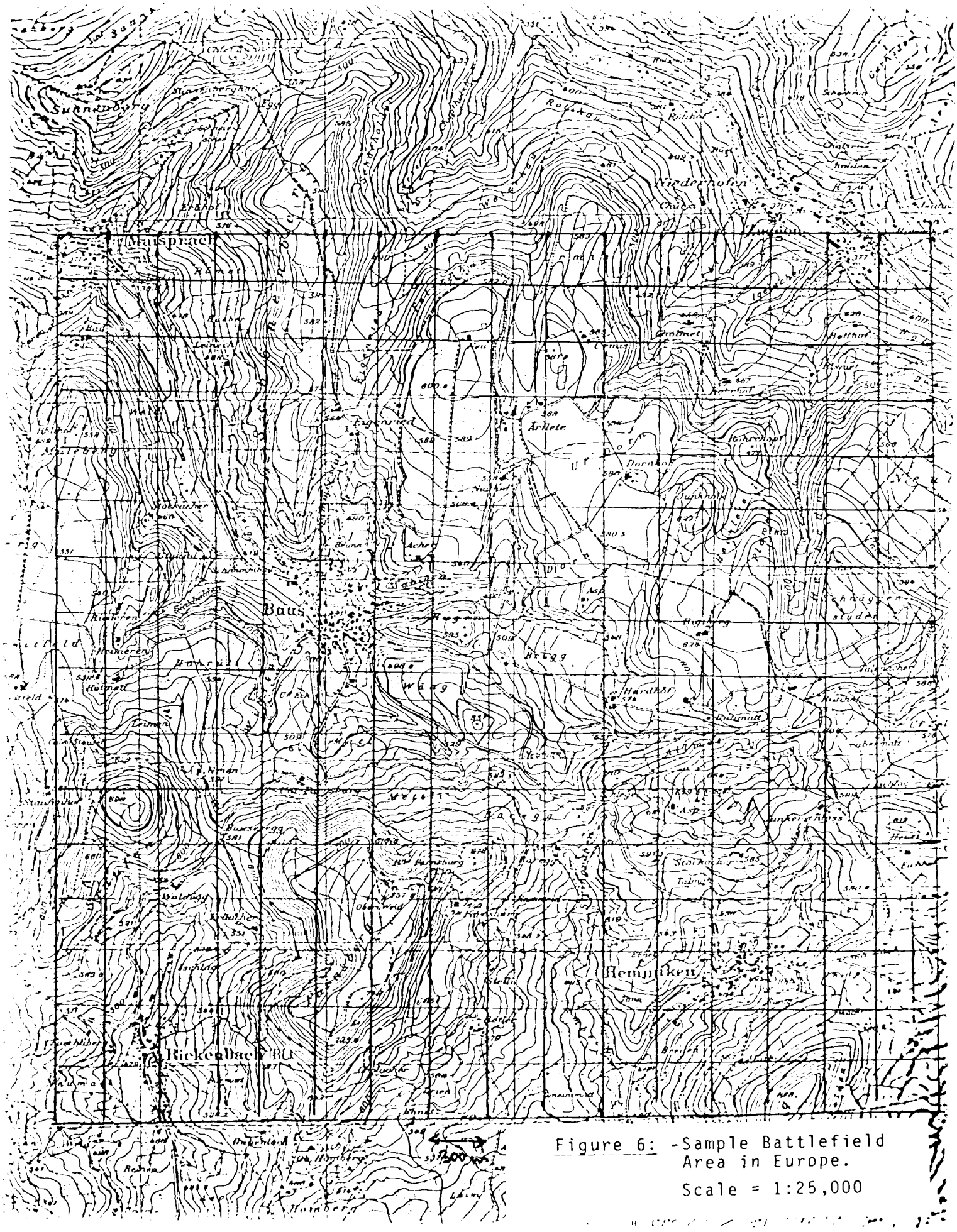


Figure 6: -Sample Battlefield Area in Europe.

Scale = 1:25,000

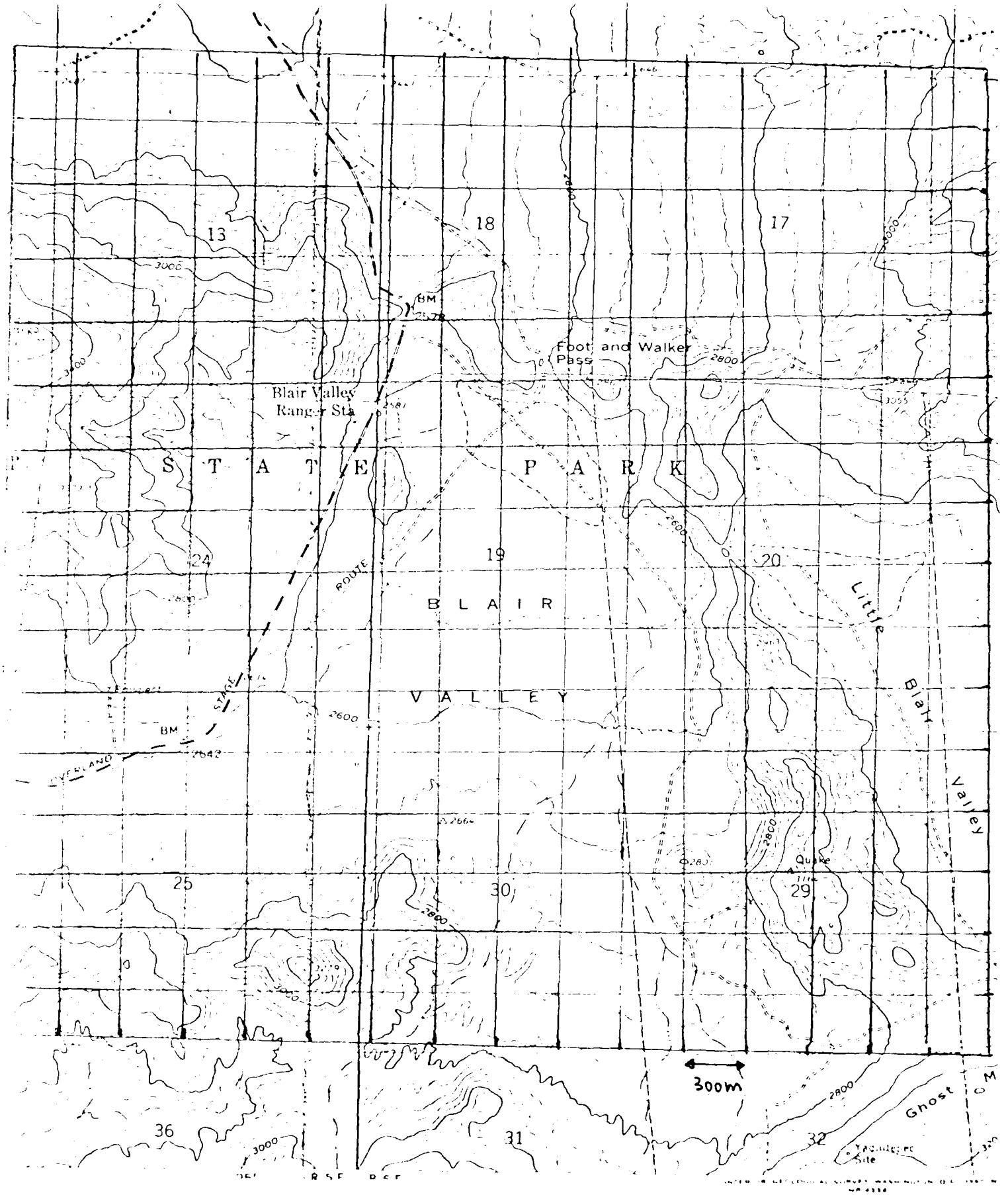


Figure 7: Sample Desert Battlefield.

Scale = 1:24,000

APPENDIX A

DETAILED CALCULATIONS FOR THE UNMODIFIED ALGORITHM

ITERATION 1

$$S = \{(U_1 + d_{1,2}), (U_1 + d_{1,5})\}$$

$0 + 2$ $\quad$ $0 + 4$

2 $\quad$ 4

$j^* = 2$

$V_2 = U_2 = 2$

2 $\quad S = \{(U_1 + d_{1,5}), (U_2 + d_{2,3}), (U_2 + d_{2,6})\}$

$0 + 4$ $\quad$ $2 + 5$ $\quad$ $2 + 1$

4 $\quad$ 7 $\quad$ 3

$j^* = 6$

$V_6 = U_6 = 3$

3 $\quad S = \{(U_1 + d_{1,5}), (U_2 + d_{2,3}), (U_6 + d_{6,5}), (U_6 + d_{6,7}), (U_6 + d_{6,10})\}$

$j^* = 5$

$0 + 4$ $\quad$ $2 + 5$ $\quad$ $3 + 5$ $\quad$ $3 + 1$ $\quad$ $3 + 9$

4 $\quad$ 7 $\quad$ 8 $\quad$ 4 $\quad$ 12

$V_5 = U_5 = 4$

4 $\quad S = \{(U_2 + d_{2,8}), (U_6 + d_{6,7}), (U_6 + d_{6,10}), (U_5 + d_{5,9})\}$

$j^* = 7$

$2 + 5$ $\quad$ $3 + 1$ $\quad$ $3 + 9$ $\quad$ $4 + 3$

7 $\quad$ 4 $\quad$ 12 $\quad$ 7

$V_7 = U_7 = 4$

5 S = $\{(U_2 + d_{2,3}), (U_6 + d_{6,10}), (U_5 + d_{5,9}), (U_7 + d_{7,3}), (U_7 + d_{7,8}), (U_7 + d_{7,11})\}$ $j^* = 3$

2 + 5 | 3 + 9 | 4 + 3 | 4 + 6 | 4 + 7 | 4 + 3 $U_3 = V_3 = 7$

7 | 12 | 7 | 10 | 11 | 7

↑

6 S = $\{(U_6 + d_{6,10}), (U_5 + d_{5,9}), (U_7 + d_{7,8}), (U_7 + d_{7,11}), (U_3 + d_{3,4})\}$ $j^* = 9$

3 + 9 | 4 + 3 | 4 + 7 | 4 + 3 | 7 + 8 $U_9 = V_9 = 7$

12 | 7 | 11 | 7 | 15

↑

7 S = $\{(U_6 + d_{6,10}), (U_7 + d_{7,8}), (U_7 + d_{7,11}), (U_3 + d_{3,4}), (U_9 + d_{9,10}), (U_9 + d_{9,13})\}$ $j^* = 11$

3 + 9 | 4 + 7 | 4 + 3 | 7 + 8 | 7 + 7 | 7 + 9 $U_{11} = V_{11} = 7$

12 | 11 | 7 | 15 | 14 | 16

↑

8 S = $\{(U_6+d_{6,10}), (U_7+d_{7,8}), (U_3+d_{3,4}), (U_9+d_{9,10}), (U_9+d_{9,13}), (U_{11}+d_{11,10}), (U_{11}+d_{11,12}), (U_{11}+d_{11,15})\}$

3+4 | 4+7 | 7+8 | 7+7 | 7+9 | 7 + 1 | 7 + 9 | 7 + 6

12 | 11 | 15 | 14 | 16 | 8 | 16 | 13

↑ $j^* = 10$

$U_{10} = V_{10} = 8$

9 $S = \{(U_7 + d_{7,8}), (U_3 + d_{3,4}), (U_9 + d_{9,13}), (U_{11} + d_{11,12}), (U_{11} + d_{11,15}), (U_{10} + d_{10,14})\}$ $j^* = 14$

4 + 7 7 + 8 7 + 9 7 + 9 7 + 6 8 + 2 $U_{14} = V_{14} =$

12 11 16 16 13 10

10 $S = \{(U_7+d_{7,8}), (U_3+d_{3,4}), (U_9+d_{9,13}), (U_{11}+d_{11,12}), (U_{11}+d_{11,15}), (U_{14}+d_{14,13}), (U_{14}+d_{14,15})\}$ $j^* = 8$

4 + 7 7 + 8 7 + 9 7 + 9 7 + 6 10 + 8 10 + 1 $U_8 = V_8 = 11$

11 15 16 16 13 18 11

11 $S = \{(U_3+d_{3,4}), (U_9+d_{9,13}), (U_{11}+d_{11,12}), (U_{11}+d_{11,15}), (U_{14}+d_{14,13}), (U_{14}+d_{14,15}), (U_8+d_{8,4}), (U_8+d_{8,12})\}$

7 + 8 7 + 9 7 + 9 7 + 6 10 + 8 10 + 1 11 + 4 11 + 3

15 16 16 13 18 11 15 14

$j^* = 15$

$U_{15} = V_{15} =$

12 $S = \{(U_3+d_{3,4}), (U_9+d_{9,13}), (U_{11}+d_{11,12}), (U_{14}+d_{14,13}), (U_8+d_{8,4}), (U_8+d_{8,12}), (U_{15}+d_{15,16})\}$

11 + 2 $j^* = 1$

15 16 16 18 15 14 13 $U_{16} = V_{16} =$

A-3

13 $S = \{(U_3+d_{3,4}),(U_9+d_{9,13}),(U_{11}+d_{11,12}),(U_{14}+d_{14,13}),(U_8+d_{8,4}),(U_8+d_{8,12})(U_{16}+d_{16,12})\}$ $j^* = 12$

$13 + 8$ $U_{12} = V_{12} =$

15 16 16 18 15 14 21

14 $S = \{(U_3 + d_{3,4}),(U_9 + d_{9,13}),(U_{14} + d_{14,3}), (U_8 + d_{8,4}\}$ $j^* = 4$

15 16 18 15

$U_4 = V_4 = 15$

15 $S = \{(U_9 + d_{9,13}),(U_{14} + d_{14,13})\}$ $j^* = 13$

16 18

$U_{13} = V_{13} =$

APPENDIX B

AN INTUITIVE "PROOF" OF THE SHORTEST PATH ALGORITHM

Consider yourself located at node i, and you want to find the shortest path to all the other nodes. First, you must move to some other node and assume you could move to nodes j, k, ℓ, or m. Assume the path from i to m is the shortest. We now postulate that the shortest path from i to m is the direct path from i to m. One might ask, "Couldn't it be that some other path passing through intermediate nodes would be shorter?" The answer is "no," because it there were, the first leg of that path would have to go through node j, k, or ℓ. But we already know that going to j is longer than going to m; the same is true for k and ℓ. Any other path to m would be longer therefore, than i-m already on its first leg.

We next investigate the paths emanating from i and m, and we find that one which moves us away from either node to a new node with a total minimum path length, excluding a move between m and i or i and m. (This reflects the fact that columns i and m have been crossed out). One or more will have minimal total length, and we select one of them. From neither node i nor node m can there be another path with total length (measured from node i) which would be shorter than the one determined above, because again, the total length of that path to the first

intermediate point would be longer than the length of the path determined above.

Therefore, nodes reached by minimizing (among the set of reachable nodes), the total distance from node i will have the property that their minimum distance from i is known.

Since the network is of finite dimensions and since at every step, so long as there are available emanating branches from nodes to which the minimum distance is known, we will eventually cover all the nodes.

One might add a side-remark here. If the problem is really limited to finding the shortest distance from a source node to a destination node, the algorithm could be terminated as soon as this destination node appears in the "solution set" (nodes for which minimum distances are known).

32-bit processor: iAPX 432 (43201/43202)

Intel Corp.
3585 S.W. 198th Ave.
Aloha, OR 97007
(503) 681-8080

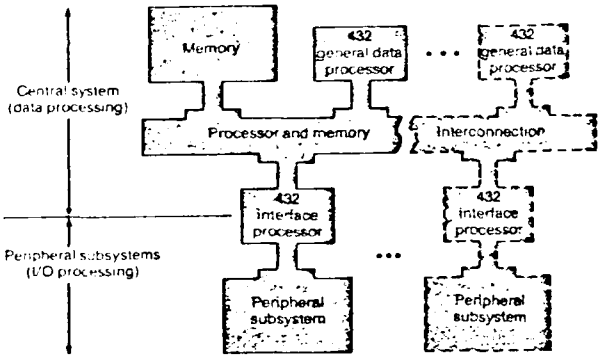
Alternative source: none

The iAPX 432 general data processor is a 32-bit microprocessor comprising two VLSI chips, the 43201 and the 43202. A third chip, the 43203, provides I/O facilities. A new object-based architecture significantly reduces the cost of large software systems while enhancing reliability and security. The processor provides 2^{40} bytes of virtual address space with capability-based addressing and protection. A functional-redundancy-checking mode allows hardware errors to be detected. Software-transparent multiprocessing allows system performance to be matched to the application and simplifies subsequent expansion. The unconventional instruction set is written in an intermediate-level language, thereby easing the writing of efficient compilers. In turn, these compilers can efficiently handle high-level languages. The machine is supported by a systems-implementation language compiler that provides a superset of Ada. Compatibility with older microprocessors is maintained because the physical hardware interface is the familiar and popular Multibus. In that way, more limited processors can play supporting roles in large systems.

Hardware		
Model	**Description**	**Price (100 qty)**
43201	iAPX 432 microinstruction sequencer	$800.00*
43202	iAPX 432 execution unit	
43203	iAPX 432 interface processor	$ 495.00*
iSBC 432/100	Evaluation board for iAPX 432	$4250.00
Intellec Series 3	Evaluation system for iAPX 432 plus additional memory	$23,300.00

*Projected cost of three-chip set in 1984 is under $300.

Specifications	
Data word size	32 bits
Instruction word size	32 bits
Physical addressing range	16 Mbytes
Virtual addressing range	1 terabyte
Number of basic instructions	221
Shortest instruction time (many)	1.25 μs
Longest instruction time (message transmission)	200 μs
Clock frequency	5/8 MHz (two versions)
Clock phases/voltage swing	2/TTL
Package	64-pin QUIP (each chip)
Power requirements	5 V/400 mA (43201) 5 V/455 mA (43202)



Comments

The instruction set of the iAPX 432 is unlike that of other microprocessors because programming is intended to be performed in a higher-level language than conventional assembly language. The 43201 functions as an instruction-decoding unit, changing macroinstructions into corresponding sets of microinstructions. The 43202 functions as a microexecution unit. There are a total of 221 instructions. To achieve efficient storage, the instructions are encoded without regard for byte, word, or other artificial boundaries. Each instruction occupies exactly the number of bits required for its complete specification. A comprehensive set of operators manipulates several hardware-recognized data types. There are eight primitive data types, divided into four classes: character, ordinal, integer, and real. The operators for these data types can be divided into several broad groups: arithmetic, logical, relational, conversion, move, and bit-field manipulation.

Software features, such as the powerful intermediate-level instruction set, are designed to improve the productivity of software development. In addition to basic instructions for operations such as data transfer, arithmetic, logical comparison, and conversion, the iAPX 432 accepts high-level instructions for operations that fall into the major categories of communication, storage allocation, mutual exclusion, and protection. Also, there are automatic operations such as process dispatching and low-level scheduling, message synchronization, and queuing. System integrity is protected by "need-to-know" addressing at the data-structure level. When processors are configured in self-checking pairs, processor hardware errors can be detected automatically. As already noted, the use of macroinstructions simplifies compiler design and boosts software productivity. Also, it allows convenient software revision as a system grows and new services evolve.

Software support includes an expanded Ada compiler. Compilers for other high-level languages are under development. Evaluation boards and compilers are intended for use with the Intellec development system. Time-sharing cross-software support is already available.

Hardware support includes a board-level, iSBC-compatible evaluation system designated the Intellec 432/100. This system includes an iSBC 432/100 board, which is Multibus-compatible and has an RS-232-C serial interface. Also included with the system is object-builder evaluation software and seven introductory texts and references. The hardware works in conjunction with an Intellec development system. The Multibus hardware is of course compatible with a broad range of board-level hardware from Intel and other manufacturers.

APPENDIX D

16-bit microprocessors: TMS99105, 99110, 99120

Texas Instruments Inc.
8600 Commerce Park Dr. (M/S6404)
Houston, TX 77036
(713) 778-6634

Alternative sources: none

The TMS99105, TMS99110, and TMS99120 are third-generation microprocessors that evolved from the original TMS9900 family. The 99105's instruction set is object-code-compatible with that of the TMS9900, augmented by extra arithmetic instructions. Also included are linkage-stack and memory-bit-test instructions. The architecture has been streamlined to provide machine and memory-bus cycle times of 167 ns—to ensure efficient memory interfacing and high throughput. The 99110 has three additional instructions to support the TIM99610 memory mapper, which addresses up to 16 Mbytes of memory. The 99120 has the same capabilities as the 99105, but also supports the Microprocessor Pascal high-level language.

Comments

The instruction set for the third-generation parts consists of the TMS9900 processor's 69 commands plus additional instructions. That raises the total number of basic instructions to 85 for the 99105 and 99120, and to 98 for the 99110. The new instructions include double-precision addition, subtraction, and shifting and signed multiplication and division. Also, the 99110 has additional instructions for memory mapping. The high-speed multiplication is performed in 3.8 μs.

Software features include the linkage-stack and memory-bit-test instructions. Through firmware resident in macrostore, the native functions of the processors can be customized by the system designer. Functions can also be changed dynamically by a system programmer. A special interface included on the chip allows additional functions to be executed by an attached processor that has a private memory system. The use of custom hardware to perform specific functions can greatly increase system throughput. The 99120 supports Microprocessor Pascal, which is a subset of the Pascal high-level language.

Software support includes currently available TMS9900 software-development programs on the AMPL development system, plus a library of component software products. Editors, assemblers, debugging routines, compilers, and high-level languages (such as Basic and Pascal) are readily available.

Hardware support consists of many of the existing TMS9900 peripheral devices (such as floppy-disk, GPIB, CRT, and video-display controllers), as well as the AMPL development system.

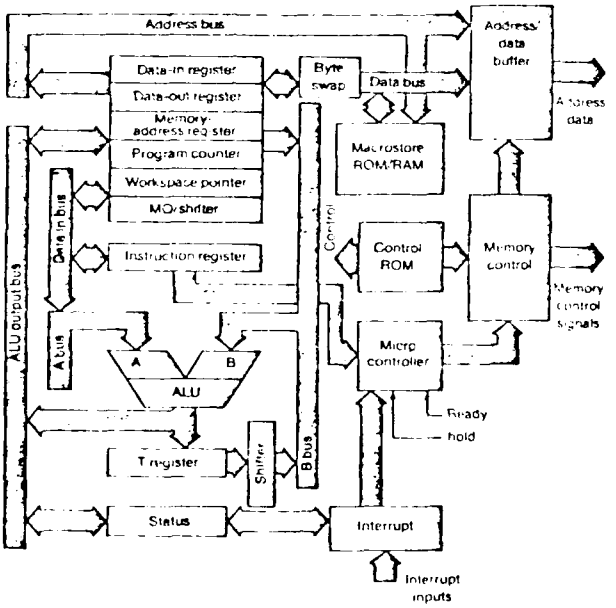
Hardware

Model	Description	Price (100 qty)
TMS99105	16-bit microprocessor	$65.00
TMS99110	16-bit microprocessor	$99.00
TMS99120	16-bit microprocessor	TBA
TIM99610	Memory mapper	$30.15
TMS9902	Asynchronous serial I/O interface	$ 4.25
TMS9903	Synchronous serial I/O interface	$15.00
TMS9909	Floppy-disk controller	$34.50
TMS9914-A	GPIB talker-listener	$23.50
TMS9918A	CRT graphics controller	$23.50
TMS9927	CRT controller	$17.00
TMAM9000	AMPL development system (single-user version)	$13,350.00
TMS99650	Processor-to-processor interface (2-port, 256 × 8 RAM)	TBA
TMS99840-41	Programmable universal peripheral controller	TBA

TBA = to be announced

Specifications

Data word size	8 or 16 bits
Address bus size	16 bits (plus 2 bits)
Direct addressing range	64 kbytes (256 kbytes)
Instruction word size	16 bits
Number of basic instructions	85 (99105, 99120) 98 (99110)
Shortest instruction time (many)	0.5 μs
Longest instruction time (floating-point addition)	100 μs
Clock frequency	24 MHz
Clock phases/voltage swing	1/TTL
Package	40-pin DIP
Power requirements	5 V/180 mA




DATE
FILMED
58